Lab

PLC-based motor control system: *Question 91 and 92*, **completed objectives due by the end of day 2, section 3**

Exam

Day 3 of next section – only a simple calculator may be used!

Specific objectives for the "mastery" exam:

- Electricity Review: Calculate voltages, currents, powers and/or resistances in a DC series-parallel circuit
- Sketch proper wire connections for sourcing or sinking PLC I/O points
- Determine status of PLC discrete output given discrete input states and a simple RLL program listing
- Calculate either the full-load current or the horsepower of an electric motor (either single- or three-phase) given the line voltage and one of the other parameters
- Solve for a specified variable in an algebraic formula
- Determine the possibility of suggested faults in a simple PLC circuit given a wiring diagram, RLL program listing, and reported symptoms
- INST240 Review: Calculate ranges for hydrostatic (DP) level-measuring instruments given physical dimensions and fluid densities
- INST250 Review: Convert between different pressure units (PSI, "W.C., bar, etc.) showing proper mathematical cancellation of units (i.e. the "unity fraction" technique)
- INST262 Review: Identify specific instrument calibration errors (zero, span, linearity, hysteresis) from data in an "As-Found" table

Recommended daily schedule

Day 1

Theory session topic: Introduction to PLCs

Questions 1 through 20; answer questions 1-10 in preparation for discussion (remainder for practice)

Day 2

Theory session topic: Contact and coil programming

Questions 21 through 40; answer questions 21-27 in preparation for discussion (remainder for practice)

Day 3

Theory session topic: Counter instructions

Questions 41 through 60; answer questions 41-47 in preparation for discussion (remainder for practice)

Day 4

Theory session topic: Counter applications

Questions 61 through 80; answer questions 61-67 in preparation for discussion (remainder for practice)

Feedback questions *(81 through 90)* are optional and may be submitted for review at the end of the day

Course Syllabus

INSTRUCTOR CONTACT INFORMATION:

Tony Kuphaldt
(360)-752-8477 [office phone]
(360)-752-7277 [fax]
tony.kuphaldt@btc.ctc.edu

DEPT/COURSE #: INST 231

CREDITS: 3 **Lecture Hours:** 10 **Lab Hours:** 50 **Work-based Hours:** 0

COURSE TITLE: PLC Programming

COURSE DESCRIPTION: In this course you will learn how to wire, program, and configure programmable logic controllers (PLCs) to perform discrete control functions including combinational logic, counters, and timers. **Pre/Corequisite course:** INST 230 (Motor Controls) **Prerequisite course:** MATH&141 (Precalculus 1) with a minimum grade of "C"

COURSE OUTCOMES: Construct, program, and efficiently diagnose control systems incorporating programmable logic controllers (PLCs).

COURSE OUTCOME ASSESSMENT: PLC wiring, programming, and configuration outcomes are ensured by measuring student performance against mastery standards, as documented in the Student Performance Objectives. Failure to meet all mastery standards by the next scheduled exam day will result in a failing grade for the course.

STUDENT PERFORMANCE OBJECTIVES:

- Without references or notes, within a limited time (3 hours total for each exam session), independently perform the following tasks. Multiple re-tries are allowed on mastery (100% accuracy) objectives, each with a different set of problems:
  → Calculate voltages, currents, powers, and/or resistances in a DC series-parallel circuit, with 100% accuracy (mastery)
  → Sketch proper wire connections for sourcing or sinking PLC I/O points given schematic or pictorial diagrams of the components, with 100% accuracy (mastery)
  → Determine status of a PLC discrete output given input states and a simple RLL program, with 100% accuracy (mastery)
  → Calculate either the full-load current or the horsepower of an electric motor (either single- or three-phase) given the line voltage and one of the other parameters
  → Solve for specified variables in algebraic formulae, with 100% accuracy (mastery)
  → Determine the possibility of suggested faults in a simple PLC circuit given measured values (voltage, current), a schematic diagram, and reported symptoms, with 100% accuracy (mastery)
  → Program a PLC to fulfill a specified control system function
- In a team environment and with full access to references, notes, and instructor assistance, perform the following tasks:
  → Demonstrate proper use of safety equipment and application of safe procedures while using power tools, and working on live systems
  → Communicate effectively with teammates to plan work, arrange for absences, and share responsibilities in completing all labwork
  → Construct and commission a motor start/stop system using a PLC as the control element
  → Generate an accurate wiring diagram compliant with industry standards documenting your team's motor control system
- Independently perform the following tasks with 100% accuracy (mastery). Multiple re-tries are allowed with different specifications/conditions each time:
  → Program a start/stop function in a PLC and wire it to control an electromechanical relay

COURSE OUTLINE: A course calendar in electronic format (Excel spreadsheet) resides on the Y: network drive, and also in printed paper format in classroom DMC130, for convenient student access. This calendar is updated to reflect schedule changes resulting from employer recruiting visits, interviews, and other impromptu events. Course worksheets provide comprehensive lists of all course assignments and activities, with the first page outlining the schedule and sequencing of topics and assignment due dates. These worksheets are available in PDF format at `http://www.ibiblio.org/kuphaldt/socratic/sinst`

- INST231 Section 1 (PLC contact, coil, and counter programming): 4 days theory and labwork
- INST231 Section 2 (PLC timer and sequence programming): 2 days theory and labwork + 1 day for mastery/proportional Exams

METHODS OF INSTRUCTION: Course structure and methods are intentionally designed to develop critical-thinking and life-long learning abilities, continually placing the student in an active rather than a passive role.

- **Independent study:** daily worksheet questions specify *reading assignments*, *problems* to solve, and *experiments* to perform in preparation (before) classroom theory sessions. Open-note quizzes and work inspections ensure accountability for this essential preparatory work. The purpose of this is to convey information and basic concepts, so valuable class time isn't wasted transmitting bare facts, and also to foster the independent research ability necessary for self-directed learning in your career.
- **Classroom sessions:** a combination of *Socratic discussion*, short *lectures*, *small-group* problem-solving, and hands-on *demonstrations/experiments* review and illuminate concepts covered in the preparatory questions. The purpose of this is to develop problem-solving skills, strengthen conceptual understanding, and practice both quantitative and qualitative analysis techniques.
- **Hands-on PLC programming challenges:** daily worksheet questions specify realistic scenarios requiring students to develop real PLC programs on their PLC trainers to implement the desired control function(s).
- **Lab activities:** an emphasis on constructing and documenting *working projects* (real instrumentation and control systems) to illuminate theoretical knowledge with practical contexts. Special projects off-campus or in different areas of campus (e.g. BTC's Fish Hatchery) are encouraged. Hands-on *troubleshooting exercises* build diagnostic skills.
- **Feedback questions:** sets of *practice problems* at the end of each course section challenge your knowledge and problem-solving ability in current as as well as first year (Electronics) subjects. These are optional assignments, counting neither for nor against your grade. Their purpose is to provide you and your instructor with direct feedback on what you have learned.

STUDENT ASSIGNMENTS/REQUIREMENTS: All assignments for this course are thoroughly documented in the following course worksheets located at:
`http://www.ibiblio.org/kuphaldt/socratic/sinst/index.html`

- `INST231_sec1.pdf`
- `INST231_sec2.pdf`

EVALUATION AND GRADING STANDARDS: (out of 100% for the course grade)

- Completion of all mastery objectives = 50%
- Mastery exam score (first attempt) = 10%
- Proportional exam score = 30%
- Lab questions = 10%
- Quiz penalty = -1% per failed quiz
- Tardiness penalty = -1% per incident (1 "free" tardy per course)
- Attendance penalty = -1% per hour (12 hours "sick time" per quarter)
- Extra credit = +5% per project (assigned by instructor based on individual learning needs)

All grades are criterion-referenced (i.e. no grading on a "curve")

100% $\geq$ **A** $\geq$ 95%	95% > **A-** $\geq$ 90%		
90% > **B+** $\geq$ 86%	86% > **B** $\geq$ 83%	83% > **B-** $\geq$ 80%	
80% > **C+** $\geq$ 76%	76% > **C** $\geq$ 73%	73% > **C-** $\geq$ 70%	(minimum passing course grade)
70% > **D+** $\geq$ 66%	66% > **D** $\geq$ 63%	63% > **D-** $\geq$ 60%	60% > **F**

A graded "preparatory" quiz at the start of each classroom session gauges your independent learning prior to the session. A graded "summary" quiz at the conclusion of each classroom session gauges your comprehension of important concepts covered during that session. If absent during part or all of a classroom session, you may receive credit by passing comparable quizzes afterward or by having your preparatory work (reading outlines, work done answering questions) thoroughly reviewed prior to the absence.

Absence on a scheduled exam day will result in a 0% score for the proportional exam unless you provide documented evidence of an unavoidable emergency.

If you fail a mastery exam, you must re-take a different version of that mastery exam on a different day. Multiple re-tries are allowed, on a different version of the exam each re-try. There is no penalty levied on your course grade for re-taking mastery exams, but failure to successfully pass a mastery exam by the due date (i.e. by the date of the *next* exam in the course sequence) will result in a failing grade (F) for the course.

If any other "mastery" objectives are not completed by their specified deadlines, your overall grade for the course will be capped at 70% (C- grade), and you will have one more school day to complete the unfinished objectives. Failure to complete those mastery objectives by the end of that extra day (except in the case of documented, unavoidable emergencies) will result in a failing grade (F) for the course.

"Lab questions" are assessed by individual questioning, at any date after the respective lab objective (mastery) has been completed by your team. These questions serve to guide your completion of each lab exercise and confirm participation of each individual student. Grading is as follows: full credit for thorough, correct answers; half credit for partially correct answers; and zero credit for major conceptual errors. All lab questions must be answered by the due date of the lab exercise.

Extra credit opportunities exist for each course, and may be assigned to students upon request. The student and the instructor will first review the student's performance on feedback questions, homework, exams, and any other relevant indicators in order to identify areas of conceptual or practical weakness. Then, both will work together to select an appropriate extra credit activity focusing on those identified weaknesses, for the purpose of strengthening the student's competence. A due date will be assigned (typically two weeks following the request), which must be honored in order for any credit to be earned from the activity. Extra credit may be denied at the instructor's discretion if the student has not invested the necessary preparatory effort to perform well (e.g. lack of preparation for daily class sessions, poor attendance, no feedback questions submitted, etc.).

REQUIRED STUDENT SUPPLIES AND MATERIALS:

- Course worksheets available for download in PDF format
- *Lessons in Industrial Instrumentation* textbook, available for download in PDF format
  → Access worksheets and book at: `http://www.ibiblio.org/kuphaldt/socratic/sinst`
- Spiral-bound notebook for reading annotation, homework documentation, and note-taking.
- Instrumentation reference CD-ROM (free, from instructor). This disk contains many tutorials and datasheets in PDF format to supplement your textbook(s).
- Tool kit (see detailed list)
- Simple scientific calculator (non-programmable, non-graphing, no unit conversions, no numeration system conversions), TI-30Xa or TI-30XIIS recommended
- Small "brick" PLC and HMI panel (*Automation Direct option*):
  → Automation Direct CLICK PLC model C0-00DD1-D (price ≈ $70) 8 discrete (DC) inputs, 6 discrete (DC) outputs
  → *or* Automation Direct CLICK PLC model C0-02DD1-D (price ≈ $130) 4 discrete (DC) inputs, 4 discrete (DC) outputs, 2 analog inputs, 2 analog outputs, RS-485 Modbus communications port, real-time clock and calendar
  → Automation Direct CLICK 24 VDC power supply model C0-00AC (price ≈ $30) 24 VDC at 0.5 amp maximum output
  → Automation Direct C-More Micro HMI panel 3 inch EA1-S3ML-N (price ≈ $150)
  → *optional* Automation Direct C-More Micro touch-screen HMI panel 3 inch EA1-S3ML (price ≈ $190)
  → Automation Direct USB/serial adapter and cable part EA-MG-PGM-CBL (price ≈ $40) necessary for programming the C-More Micro HMI panel (also works for programming the PLC)
  → **Note:** We have found the Autmoation Direct software works equally well through a 9-pin serial port as through a USB port (with converter), and is very "friendly" to use.
- Small "brick" PLC and HMI panel (*Allen-Bradley option*):
  → Rockwell (Allen-Bradley) MicroLogix 1000 model 1761-L10BWA (price ≈ $85 with BTC student discount at North Coast Electric) 6 discrete (DC) inputs, 4 discrete (relay) outputs
  → *or* Rockwell (Allen-Bradley) MicroLogix 1100 model 1763-L16BWA (price ≈ $240 with BTC student discount at North Coast Electric) 10 discrete (DC) inputs, 6 discrete (DC) outputs, 2 analog inputs, RS-485 communication port, 10 Mbit/s Ethernet communication port, embedded web server for remote monitoring of data points (series A or B programmable using free MicroLogix Lite software)
  → Rockwell (Allen-Bradley) cable part 1761-CBL-PM02 (price ≈ $30 with BTC student discount at North Coast Electric)
  → Automation Direct C-More Micro HMI panel 3 inch EA1-S3ML-N (price ≈ $150)
  → *optional* Automation Direct C-More Micro touch-screen HMI panel 3 inch EA1-S3ML (price ≈ $190)
  → Automation Direct cable part EA-MLOGIX-CBL (price ≈ $30) and adapter part EA-MG-SP1 (price ≈ $50) necessary for connecting the C-More Micro HMI panel to an Allen-Bradley MicroLogix 1000 PLC
  → Automation Direct USB/serial adapter and cable part EA-MG-PGM-CBL (price ≈ $40) necessary for programming the C-More Micro HMI panel
  → **Note:** Programming Allen-Bradley PLCs is best done using a PC with a 9-pin serial port. We have found trying to use a USB-to-serial adapter very troublesome with Allen-Bradley software!

ADDITIONAL INSTRUCTIONAL RESOURCES:

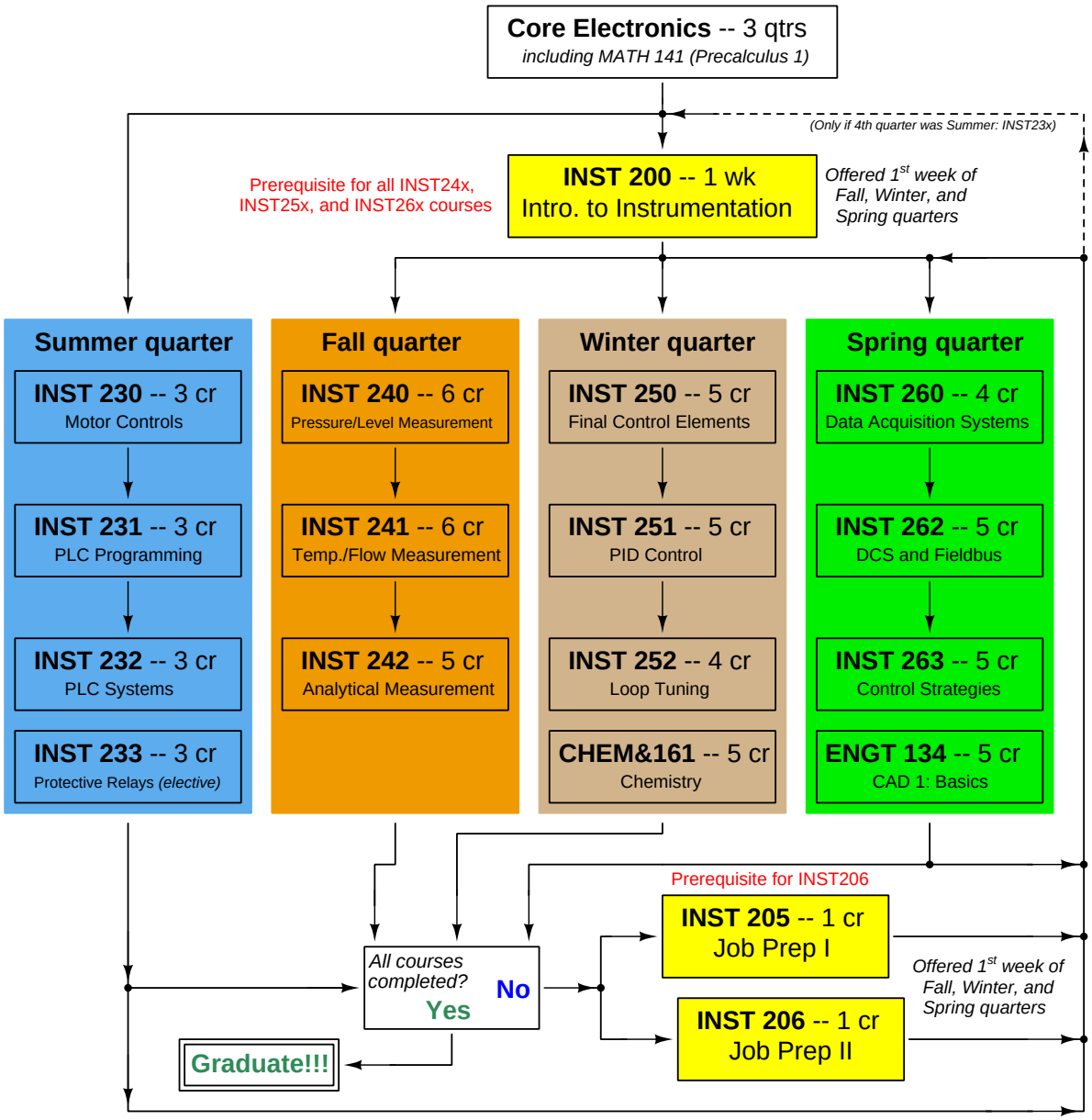
- The BTC Library hosts a substantial collection of textbooks and references on the subject of Instrumentation, as well as links in its online catalog to free Instrumentation e-book resources available on the Internet.
- "BTCInstrumentation" channel on YouTube (`http://www.youtube.com/BTCInstrumentation`), hosts a variety of short video tutorials and demonstrations on instrumentation.
- ISA Student Section at BTC meets regularly to set up industry tours, raise funds for scholarships, and serve as a general resource for Instrumentation students. Membership in the ISA is $10 per year, payable to the national ISA organization. Membership includes a complementary subscription to *InTech* magazine.
- ISA website (`http://www.isa.org`) provides all of its standards in electronic format, many of which are freely available to ISA members.
- *Cad Standard* (CadStd) or similar AutoCAD-like drafting software (useful for sketching loop and wiring diagrams). Cad Standard is a simplified clone of AutoCAD, and is freely available at: `http://www.cadstd.com`

CAMPUS EMERGENCIES: If an emergency arises, your instructor may inform you of actions to follow. You are responsible for knowing emergency evacuation routes from your classroom. If police or university officials order you to evacuate, do so calmly and assist those needing help. You may receive emergency information alerts via the building enunciation system, text message, email, or BTC's webpage (`http://www.btc.ctc.edu`), Facebook or Twitter. Refer to the emergency flipchart in the lab room (located on the main control panel) for more information on specific types of emergencies.

ACCOMMODATIONS: If you think you could benefit from classroom accommodations for a disability (physical, mental, emotional, or learning), please contact our Accessibility Resources office. Call (360)-752-8345, email `ar@btc.ctc.edu`, or stop by the AR Office in the Admissions and Student Resource Center (ASRC), Room 106, College Services Building

file `INST231syllabus`

Sequence of second-year Instrumentation courses



The particular sequence of courses you take during the second year depends on when you complete all first-year courses and enter the second year. Since students enter the second year of Instrumentation at four different times (beginnings of Summer, Fall, Winter, and Spring quarters), the particular course sequence for any student will likely be different from the course sequence of classmates.

Some second-year courses are only offered in particular quarters with those quarters not having to be in sequence, while others are offered three out of the four quarters and must be taken in sequence. The following layout shows four typical course sequences for second-year Instrumentation students, depending on when they first enter the second year of the program:

Possible course schedules depending on date of entry into 2nd year

file sequence

General Values and Expectations

Success in this career requires: professional integrity, resourcefulness, persistence, close attention to detail, and intellectual curiosity. Poor judgment spells disaster in this career, which is why employer background checks (including social media and criminal records) and drug testing are common. The good news is that character and clear thinking are malleable traits: unlike intelligence, these qualities can be acquired and improved with effort. *This is what you are in school to do* – increase your "human capital" which is the sum of all knowledge, skills, and traits valuable in the marketplace.

Mastery: You must master the fundamentals of your chosen profession. "Mastery" assessments challenge you to demonstrate 100% competence (with multiple opportunities to re-try). Failure to complete any mastery objective(s) by the deadline date caps your grade at a C−. Failure to complete by the end of the next school day results in a failing (F) grade.

Punctuality and Attendance: You are expected to arrive on time and be "on-task" all day just as you would for a job. Each student has 12 hours of "sick time" per quarter applicable to absences not verifiably employment-related, school-related, weather-related, or required by law. Each student must confer with the instructor to apply these hours to any missed time – this is not done automatically. Students may donate unused "sick time" to whomever they specifically choose. You must contact your instructor and lab team members immediately if you know you will be late or absent or must leave early. Absence on an exam day will result in a zero score for that exam, unless due to a documented emergency.

Time Management: You are expected to budget and prioritize your time, just as you will be on the job. You will need to reserve enough time outside of school to complete homework, and strategically apply your time during school hours toward limited resources (e.g. lab equipment). Frivolous activities (e.g. games, social networking, internet surfing) are unacceptable when work is unfinished. Trips to the cafeteria for food or coffee, smoke breaks, etc. must not interfere with team participation.

Independent Study: This career is marked by continuous technological development and ongoing change, which is why *self-directed learning* is ultimately more important to your future success than specific knowledge. To acquire and hone this skill, all second-year Instrumentation courses follow an "inverted" model where lecture is replaced by independent study, and class time is devoted to addressing your questions and demonstrating your learning. Most students require a *minimum* of 3 hours daily study time outside of school. Arriving unprepared (e.g. homework incomplete) is unprofessional and counter-productive. Question 0 of every worksheet lists practical study tips.

Independent Problem-Solving: The best instrument technicians are versatile problem-solvers. General problem-solving is arguably the most valuable skill you can possess for this career, and it can only be built through persistent effort. This is why you must take every reasonable measure to *solve problems on your own* before seeking help. It is okay to be perplexed by an assignment, but you are expected to apply problem-solving strategies given to you (see Question 0) and to precisely identify where you are confused so your instructor will be able to offer targeted help. Asking classmates to solve problems for you is folly – this includes having others break the problem down into simple steps. The point is to learn how to *think on your own*. When troubleshooting systems in lab you are expected to run diagnostic tests (e.g. using a multimeter instead of visually seeking circuit faults), as well as consult the equipment manual(s) before seeking help.

Initiative: No single habit predicts your success or failure in this career better than personal initiative, which is why your instructor will demand *you do for yourself rather than rely on others to do for you.* Examples include setting up and using your BTC email account to communicate with your instructor(s), consulting manuals for technical information before asking for help, regularly checking the course calendar and assignment deadlines, avoiding procrastination, fixing small problems before they become larger problems, etc. If you find your performance compromised by poor understanding of prior course subjects, re-read those textbook sections and use the practice materials made available to you on the Socratic Instrumentation website – don't wait for anyone else to diagnose your need and offer help.

Safety: You are expected to work safely in the lab just as you will be on the job. This includes wearing proper attire (safety glasses and closed-toed shoes in the lab at all times), implementing lock-out/tag-out procedures when working on circuits with exposed conductors over 30 volts, using ladders to access elevated locations, and correctly using all tools. If you need to use an unfamiliar tool, see the instructor for directions.

Orderliness: You are expected to keep your work area clean and orderly just as you will be on the job. This includes discarding trash and returning tools at the end of every lab session, and participating in all scheduled lab clean-up sessions. If you identify failed equipment in the lab, label that equipment with a detailed description of its symptoms.

Teamwork: You will work in instructor-assigned teams to complete lab assignments, just as you will work in teams to complete complex assignments on the job. As part of a team, you must keep your teammates informed of your whereabouts in the event you must step away from the lab or will be absent for *any* reason. Any student regularly compromising team performance through lack of participation, absence, tardiness, disrespect, or other disruptive behavior(s) will be removed from the team and required to complete all labwork individually for the remainder of the quarter. The same is true for students found relying on teammates to do their work for them.

Cooperation: The structure of these courses naturally lends itself to cooperation between students. Working together, students significantly impact each others' learning. You are expected to take this role seriously, offering real help when needed and not absolving classmates of their responsibility to think for themselves or to do their own work. Solving problems for classmates and/or explaining to them what they can easily read on their own is unacceptable because these actions circumvent learning. The best form of help you can give to your struggling classmates is to share with them your tips on independent learning and problem-solving, for example *asking questions* leading to solutions rather than simply providing solutions for them.

Grades: Employers prize trustworthy, hard working, knowledgeable, resourceful problem-solvers. The grade you receive in any course is but a *partial* measure of these traits. What matters most are the traits themselves, which is why your instructor maintains detailed student records (including individual exam scores, attendance, tardiness, and behavioral comments) and will share these records with employers if you have signed the FERPA release form. You are welcome to see your records at any time, and to compare calculated grades with your own records (i.e. the grade spreadsheet available to all students). You should expect employers to scrutinize your records on attendance and character, and also challenge you with technical questions when considering you for employment.

Representation: You are an ambassador for this program. Your actions, whether on tours, during a jobshadow or internship, or while employed, can open or shut doors of opportunity for other students. Most of the job opportunities open to you as a BTC graduate were earned by the good work of previous graduates, and as such you owe them a debt of gratitude. Future graduates depend on you to do the same.

Responsibility For Actions: If you lose or damage college property (e.g. lab equipment), you must find, repair, or help replace it. If you represent BTC poorly to employers (e.g. during a tour or an internship), you must make amends. The general rule here is this: *"If you break it, you fix it!"*

Non-negotiable terms: disciplinary action, up to and including immediate failure of a course, will result from academic dishonesty (e.g. cheating, plagiarism), willful safety violations, theft, harassment, intoxication, destruction of property, or willful disruption of the learning (work) environment. Such offenses are grounds for immediate termination in this career, and as such will not be tolerated here.

file `expectations`

General tool and supply list

Wrenches

- Combination (box- and open-end) wrench set, 1/4" to 3/4" – *the most important wrench sizes are 7/16", 1/2", 9/16", and 5/8"; get these immediately!*
- Adjustable wrench, 6" handle (sometimes called "Crescent" wrench)
- Hex wrench ("Allen" wrench) set, fractional – 1/16" to 3/8"
- *Optional:* Hex wrench ("Allen" wrench) set, metric – 1.5 mm to 10 mm
- *Optional:* Miniature combination wrench set, 3/32" to 1/4" (sometimes called an "ignition wrench" set)

Note: *when turning any threaded fastener, one should choose a tool engaging the maximum amount of surface area on the fastener's head in order to reduce stress on that fastener. (e.g. Using box-end wrenches instead of adjustable wrenches; using the proper size and type of screwdriver; never using any tool that mars the fastener such as pliers or vise-grips unless absolutely necessary.)*

Pliers

- Needle-nose pliers
- Tongue-and-groove pliers (sometimes called "Channel-lock" pliers)
- Diagonal wire cutters (sometimes called "dikes")

Screwdrivers

- Slotted, 1/8" and 1/4" shaft
- Phillips, #1 and #2
- Jeweler's screwdriver set
- *Optional:* Magnetic multi-bit screwdriver (e.g. Klein Tools model 70035)

Electrical

- Multimeter, Fluke model 87-IV or better
- Alligator-clip jumper wires
- Soldering iron (10 to 40 watt) and rosin-core solder
- Resistor, potentiometer, diode assortments (from first-year lab kits)
- Package of insulated compression-style fork terminals (14 to 18 AWG wire size, #10 stud size)
- Wire strippers/terminal crimpers for 10 AWG to 18 AWG wire and insulated terminals
- *Optional:* ratcheting terminal crimp tool (e.g. Paladin 1305, Ferrules Direct FDT10011, or equivalent)

Safety

- Safety glasses or goggles (available at BTC bookstore)
- Earplugs (available at BTC bookstore)

Miscellaneous

- Simple scientific calculator (non-programmable, non-graphing, no conversions), TI-30Xa or TI-30XIIS recommended. Required for some exams!
- Masking tape (for making temporary labels)
- Permanent marker pen
- Teflon pipe tape
- Utility knife
- Tape measure, 12 feet minimum
- Flashlight

An inexpensive source of tools is your local pawn shop. Look for tools with unlimited lifetime guarantees (e.g. *Sears* "Craftsman" brand). Check for BTC student discounts as well!

file `tools`

Methods of instruction

This course develops self-instructional and diagnostic skills by placing students in situations where they are required to research and think independently. In all portions of the curriculum, the goal is to avoid a passive learning environment, favoring instead *active engagement* of the learner through reading, reflection, problem-solving, and experimental activities. The curriculum may be roughly divided into two portions: *theory* and *practical*.

Theory

In the theory portion of each course, students independently research subjects *prior* to entering the classroom for discussion. This means working through all the day's assigned questions as completely as possible. This usually requires a fair amount of technical reading, and may also require setting up and running simple experiments. At the start of the classroom session, the instructor will check each student's preparation with a quiz. Students then spend the rest of the classroom time working in groups and directly with the instructor to *thoroughly* answer all questions assigned for that day, articulate problem-solving strategies, and to approach the questions from multiple perspectives. To put it simply: fact-gathering happens outside of class and is the individual responsibility of each student, so that class time may be devoted to the more complex tasks of critical thinking and problem solving where the instructor's attention is best applied.

Classroom theory sessions usually begin with either a brief Q&A discussion or with a "Virtual Troubleshooting" session where the instructor shows one of the day's diagnostic question diagrams while students propose diagnostic tests and the instructor tells those students what the test results would be given some imagined ("virtual") fault scenario, writing the test results on the board where all can see. The students then attempt to identify the nature and location of the fault, based on the test results.

Each student is free to leave the classroom when they have completely worked through all problems and have answered a "summary" quiz designed to gauge their learning during the theory session. If a student finishes ahead of time, they are free to leave, or may help tutor classmates who need extra help.

The express goal of this "inverted classroom" teaching methodology is to help each student cultivate critical-thinking and problem-solving skills, and to sharpen their abilities as independent learners. While this approach may be very new to you, it is more realistic and beneficial to the type of work done in instrumentation, where critical thinking, problem-solving, and independent learning are "must-have" skills.

Lab

In the lab portion of each course, students work in teams to install, configure, document, calibrate, and troubleshoot working instrument loop systems. Each lab exercise focuses on a different type of instrument, with a eight-day period typically allotted for completion. An ordinary lab session might look like this:

(1) Start of practical (lab) session: announcements and planning
   (a) The instructor makes general announcements to all students
   (b) The instructor works with team to plan that day's goals, making sure each team member has a clear idea of what they should accomplish
(2) Teams work on lab unit completion according to recommended schedule:
   (First day) Select and bench-test instrument(s)
   (One day) Connect instrument(s) into a complete loop
   (One day) Each team member drafts their own loop documentation, inspection done as a team (with instructor)
   (One or two days) Each team member calibrates/configures the instrument(s)
   (Remaining days, up to last) Each team member troubleshoots the instrument loop
(3) End of practical (lab) session: debriefing where each team reports on their work to the whole class

Troubleshooting assessments must meet the following guidelines:

- Troubleshooting must be performed *on a system the student did not build themselves.* This forces students to rely on another team's documentation rather than their own memory of how the system was built.
- Each student must individually demonstrate proper troubleshooting technique.
- Simply finding the fault is not good enough. Each student must consistently demonstrate sound reasoning while troubleshooting.
- If a student fails to properly diagnose the system fault, they must attempt (as many times as necessary) with different scenarios until they do, reviewing any mistakes with the instructor after each failed attempt.

file `instructional`

Distance delivery methods

Sometimes the demands of life prevent students from attending college 6 hours per day. In such cases, there exist alternatives to the normal 8:00 AM to 3:00 PM class/lab schedule, allowing students to complete coursework in non-traditional ways, at a "distance" from the college campus proper.

For such "distance" students, the same worksheets, lab activities, exams, and academic standards still apply. Instead of working in small groups and in teams to complete theory and lab sections, though, students participating in an alternative fashion must do all the work themselves. Participation via teleconferencing, video- or audio-recorded small-group sessions, and such is encouraged and supported.

There is no recording of hours attended or tardiness for students participating in this manner. The pace of the course is likewise determined by the "distance" student. Experience has shown that it is a benefit for "distance" students to maintain the same pace as their on-campus classmates whenever possible.

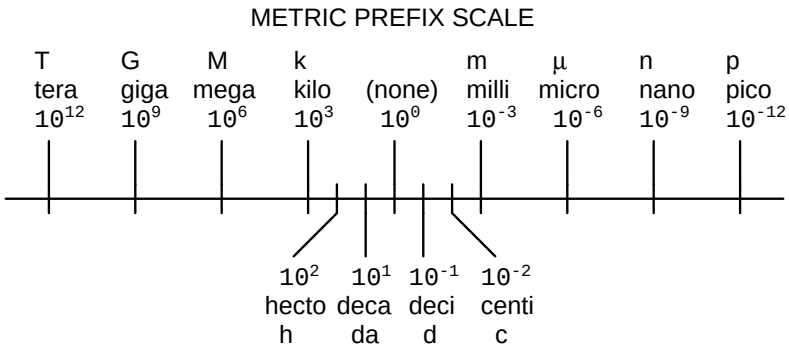
In lieu of small-group activities and class discussions, comprehension of the theory portion of each course will be ensured by completing and submitting detailed answers for *all* worksheet questions, not just passing daily quizzes as is the standard for conventional students. The instructor will discuss any incomplete and/or incorrect worksheet answers with the student, and ask that those questions be re-answered by the student to correct any misunderstandings before moving on.

Labwork is perhaps the most difficult portion of the curriculum for a "distance" student to complete, since the equipment used in Instrumentation is typically too large and expensive to leave the school lab facility. "Distance" students must find a way to complete the required lab activities, either by arranging time in the school lab facility and/or completing activities on equivalent equipment outside of school (e.g. at their place of employment, if applicable). Labwork completed outside of school must be validated by a supervisor and/or documented via photograph or videorecording.

Conventional students may opt to switch to "distance" mode at any time. This has proven to be a benefit to students whose lives are disrupted by catastrophic events. Likewise, "distance" students may switch back to conventional mode if and when their schedules permit. Although the existence of alternative modes of student participation is a great benefit for students with challenging schedules, it requires a greater investment of time and a greater level of self-discipline than the traditional mode where the student attends school for 6 hours every day. No student should consider the "distance" mode of learning a way to have more free time to themselves, because they will actually spend more time engaged in the coursework than if they attend school on a regular schedule. It exists merely for the sake of those who cannot attend during regular school hours, as an alternative to course withdrawal.

file `distance`

- **Metric prefixes**
- Yotta = 10^{24} Symbol: Y
- Zeta = 10^{21} Symbol: Z
- Exa = 10^{18} Symbol: E
- Peta = 10^{15} Symbol: P
- Tera = 10^{12} Symbol: T
- Giga = 10^{9} Symbol: G
- Mega = 10^{6} Symbol: M
- Kilo = 10^{3} Symbol: k
- Hecto = 10^{2} Symbol: h
- Deca = 10^{1} Symbol: da
- Deci = 10^{-1} Symbol: d
- Centi = 10^{-2} Symbol: c
- Milli = 10^{-3} Symbol: m
- Micro = 10^{-6} Symbol: μ
- Nano = 10^{-9} Symbol: n
- Pico = 10^{-12} Symbol: p
- Femto = 10^{-15} Symbol: f
- Atto = 10^{-18} Symbol: a
- Zepto = 10^{-21} Symbol: z
- Yocto = 10^{-24} Symbol: y



- **Conversion formulae for temperature**
- ^{o}F = (^{o}C)(9/5) + 32
- ^{o}C = (^{o}F - 32)(5/9)
- ^{o}R = ^{o}F + 459.67
- K = ^{o}C + 273.15

Conversion equivalencies for distance

1 inch (in) = 2.540000 centimeter (cm)

1 foot (ft) = 12 inches (in)

1 yard (yd) = 3 feet (ft)

1 mile (mi) = 5280 feet (ft)

Conversion equivalencies for volume

1 gallon (gal) = 231.0 cubic inches (in^3) = 4 quarts (qt) = 8 pints (pt) = 128 fluid ounces (fl. oz.) = 3.7854 liters (l)

1 milliliter (ml) = 1 cubic centimeter (cm^3)

Conversion equivalencies for velocity

1 mile per hour (mi/h) = 88 feet per minute (ft/m) = 1.46667 feet per second (ft/s) = 1.60934 kilometer per hour (km/h) = 0.44704 meter per second (m/s) = 0.868976 knot (knot – international)

Conversion equivalencies for mass

1 pound (lbm) = 0.45359 kilogram (kg) = 0.031081 slugs

Conversion equivalencies for force

1 pound-force (lbf) = 4.44822 newton (N)

Conversion equivalencies for area

1 acre = 43560 square feet (ft^2) = 4840 square yards (yd^2) = 4046.86 square meters (m^2)

Conversion equivalencies for common pressure units (either all gauge or all absolute)

1 pound per square inch (PSI) = 2.03602 inches of mercury (in. Hg) = 27.6799 inches of water (in. W.C.) = 6.894757 kilo-pascals (kPa) = 0.06894757 bar

1 bar = 100 kilo-pascals (kPa) = 14.504 pounds per square inch (PSI)

Conversion equivalencies for absolute pressure units (only)

1 atmosphere (Atm) = 14.7 pounds per square inch absolute (PSIA) = 101.325 kilo-pascals absolute (kPaA) = 1.01325 bar (bar) = 760 millimeters of mercury absolute (mmHgA) = 760 torr (torr)

Conversion equivalencies for energy or work

1 british thermal unit (Btu – "International Table") = 251.996 calories (cal – "International Table") = 1055.06 joules (J) = 1055.06 watt-seconds (W-s) = 0.293071 watt-hour (W-hr) = 1.05506 x 10^{10} ergs (erg) = 778.169 foot-pound-force (ft-lbf)

Conversion equivalencies for power

1 horsepower (hp – 550 ft-lbf/s) = 745.7 watts (W) = 2544.43 british thermal units per hour (Btu/hr) = 0.0760181 boiler horsepower (hp – boiler)

Acceleration of gravity (free fall), Earth standard

9.806650 meters per second per second (m/s^2) = 32.1740 feet per second per second (ft/s^2)

Physical constants

Speed of light in a vacuum (c) = 2.9979×10^8 meters per second (m/s) = 186,281 miles per second (mi/s)

Avogadro's number (N_A) = 6.022×10^{23} per mole (mol^{-1})

Electronic charge (e) = 1.602×10^{-19} Coulomb (C)

Boltzmann's constant (k) = 1.38×10^{-23} Joules per Kelvin (J/K)

Stefan-Boltzmann constant (σ) = 5.67×10^{-8} Watts per square meter-Kelvin^4 ($W/m^2 \cdot K^4$)

Molar gas constant (R) = 8.314 Joules per mole-Kelvin (J/mol-K)

Properties of Water

Freezing point at sea level = $32^oF = 0^oC$

Boiling point at sea level = $212^oF = 100^oC$

Density of water at 4^oC = 1000 kg/m^3 = 1 g/cm^3 = 1 kg/liter = 62.428 lb/ft^3 = 1.94 $slugs/ft^3$

Specific heat of water at 14^oC = 1.00002 calories/g·^oC = 1 BTU/lb·^oF = 4.1869 Joules/g·^oC

Specific heat of ice $\approx$ 0.5 calories/g·^oC

Specific heat of steam $\approx$ 0.48 calories/g·^oC

Absolute viscosity of water at 20^oC = 1.0019 centipoise (cp) = 0.0010019 Pascal-seconds (Pa·s)

Surface tension of water (in contact with air) at 18^oC = 73.05 dynes/cm

pH of pure water at 25^o C = 7.0 (*pH scale = 0 to 14*)

Properties of Dry Air at sea level

Density of dry air at 20^oC and 760 torr = 1.204 mg/cm^3 = 1.204 kg/m^3 = 0.075 lb/ft^3 = 0.00235 $slugs/ft^3$

Absolute viscosity of dry air at 20^oC and 760 torr = 0.018 centipoise (cp) = 1.8×10^{-5} Pascal-seconds (Pa·s)

file `conversion_constants`

Question 0

How to get the most out of academic reading:

- Articulate your thoughts as you read (i.e. "have a conversation" with the author). This will develop *metacognition*: active supervision of your own thoughts. Write your thoughts as you read, noting points of agreement, disagreement, confusion, epiphanies, and connections between different concepts or applications. These notes should also document important math formulae, explaining in your own words what each formula means and the proper units of measurement used.
- Outline, don't highlight! Writing your own summary or outline is a far more effective way to comprehend a text than simply underlining and highlighting key words. A suggested ratio is one sentence of your own thoughts per paragraph of text read. Note points of disagreement or confusion to explore later.
- Work through all mathematical exercises shown within the text, to ensure you understand all the steps.
- Imagine explaining concepts you've just learned to someone else. Teaching forces you to distill concepts to their essence, thereby clarifying those concepts, revealing assumptions, and exposing misconceptions. Your goal is to create the simplest explanation that is still technically accurate.
- Write your own questions based on what you read, as though you are a teacher preparing to test students' comprehension of the subject matter.

How to effectively problem-solve and troubleshoot:

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work. In mathematical problem-solving this means being able to identify the practical meaning (and units of measurement) of every intermediate calculation. In other words, *every step of your solution should make logical sense.*
- Sketch a diagram to help visualize the problem. When building a real system, always prototype it on paper and analyze its function *before* constructing it.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults.
- Simplify the problem and solve that simplified problem to identify strategies applicable to the original problem (e.g. change quantitative to qualitative, or visa-versa; substitute easier numerical values; eliminate confusing details; add details to eliminate unknowns; consider simple limiting cases; apply an analogy). Often you can add or remove components in a malfunctioning system to simplify it as well and better identify the nature and location of the problem.
- Work "backward" from a hypothetical solution to a new set of given conditions.

How to create more time for study:

- Kill your television and video games. Seriously – these are incredible wastes of time. Eliminate distractions (e.g. cell phone, internet, socializing) in your place and time of study.
- Use your "in between" time productively. Don't leave campus for lunch. Arrive to school early. If you finish your assigned work early, begin studying the next day's material.

Above all, cultivate persistence. Persistent effort is necessary to master anything non-trivial. The keys to persistence are (1) having the desire to achieve that mastery, and (2) realizing challenges are normal and not an indication of something gone wrong. A common error is to equate *easy* with *effective*: students often believe learning should be easy if everything is done right. The truth is that mastery never comes easy!

file `question0`

Creative Commons License

This worksheet is licensed under the **Creative Commons Attribution License**, version 1.0. To view a copy of this license, visit `http://creativecommons.org/licenses/by/1.0/` or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA. The terms and conditions of this license allow for free copying, distribution, and/or modification of all licensed works by the general public.

Simple explanation of Attribution License:

The licensor (Tony Kuphaldt) permits others to copy, distribute, display, and otherwise use this work. In return, licensees must give the original author(s) credit. For the full license text, please visit `http://creativecommons.org/licenses/by/1.0/` on the internet.

More detailed explanation of Attribution License:

Under the terms and conditions of the Creative Commons Attribution License, you may make freely use, make copies, and even modify these worksheets (and the individual "source" files comprising them) without having to ask me (the author and licensor) for permission. The one thing you must do is properly credit my original authorship. Basically, this protects my efforts against plagiarism without hindering the end-user as would normally be the case under full copyright protection. This gives educators a great deal of freedom in how they might adapt my learning materials to their unique needs, removing all financial and legal barriers which would normally hinder if not prevent creative use.

Nothing in the License prohibits the sale of original or adapted materials by others. You are free to copy what I have created, modify them if you please (or not), and then sell them at any price. Once again, the only catch is that you must give proper credit to myself as the original author and licensor. Given that these worksheets will be continually made available on the internet for free download, though, few people will pay for what you are selling unless you have somehow added value.

Nothing in the License prohibits the application of a more restrictive license (or no license at all) to derivative works. This means you can add your own content to that which I have made, and then exercise full copyright restriction over the new (derivative) work, choosing not to release your additions under the same free and open terms. An example of where you might wish to do this is if you are a teacher who desires to add a detailed "answer key" for your own benefit but *not* to make this answer key available to anyone else (e.g. students).

Note: the text on this page is not a license. It is simply a handy reference for understanding the Legal Code (the full license) - it is a human-readable expression of some of its key terms. Think of it as the user-friendly interface to the Legal Code beneath. This simple explanation itself has no legal value, and its contents do not appear in the actual license.

file `license`

Questions

Question 1

Read and outline the introduction and "PLC Examples" sections of the "Programmable Logic Controllers" chapter in your *Lessons In Industrial Instrumentation* textbook. Note the page numbers where important illustrations, photographs, equations, tables, and other relevant details are found. Prepare to thoughtfully discuss with your instructor and classmates the concepts and examples explored in this reading.

file i00460

Question 2

Read selected portions of the Siemens "SIMATIC S7-200 Programmable Controller System Manual" (document A5E00307987-04, August 2008) and answer the following questions:

Locate the section discussing the PLC's *scan cycle* and describe the sequence of operations conducted by the PLC on an ongoing basis.

Locate the section discussing the PLC's memory types ("Permanent Memory" versus "Retentive Data Memory" and such), and describe the functions of each.

A very important aspect to learn about any PLC is how to specify various locations within its memory. Each manufacturer and model of PLC has its own way of "addressing" memory locations, analogous to the different ways each postal system within each country of the world specifies its mailing addresses. Locate the section of the manual discussing addressing conventions ("Accessing the Data of the S7-200"), and then answer these questions:

Identify the proper address notation for a particular bit in the Siemens PLC's memory: bit number *4* of byte *1* within the *process-image input register*.

Identify the proper address notation for a particular bit in the Siemens PLC's memory: bit number *2* of byte *0* within the *process-image output register*.

Identify the proper address notation for a "double word" of data in the Siemens PLC's memory beginning at byte *105* within the *variable memory area*. How many bits are contained in a double word?

file i03605

Question 3

Read selected portions of the Allen-Bradley "MicroLogix 1000 Programmable Controllers (Bulletin 1761 Controllers)" user manual (document 1761-6.3, July 1998) and answer the following questions:

Locate the section discussing the PLC's *operating cycle* – otherwise known as a "scan" cycle – and describe the sequence of operations conducted by the PLC on an ongoing basis.

Locate the section discussing the PLC's memory types (EEPROM and RAM), and describe the functions of each.

A very important aspect to learn about any PLC is how to specify various locations within its memory. Each manufacturer and model of PLC has its own way of "addressing" memory locations, analogous to the different ways each postal system within each country of the world specifies its mailing addresses. Locate the section of the manual discussing addressing conventions ("Addressing Data Files"), and then answer these questions:

Identify the proper address notation for a particular bit in the Allen-Bradley PLC's memory: bit number *4* of element *1* within the *input file*.

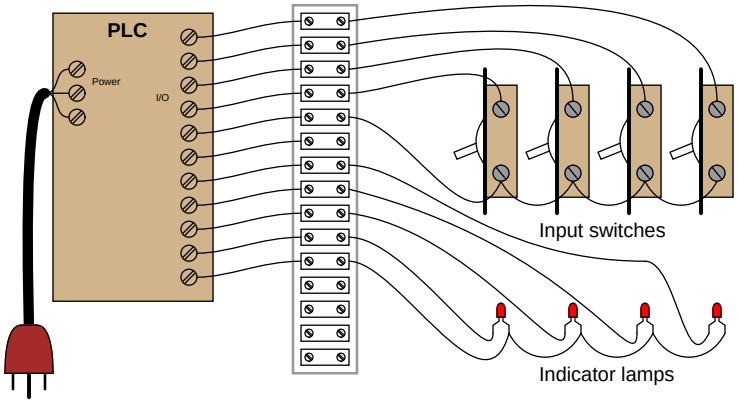
Identify the proper address notation for a particular bit in the Allen-Bradley PLC's memory: bit number *2* of element *0* within the *output file*.

Identify the proper address notation for a "word" of data in the Allen-Bradley PLC's memory: the *accumulator* word (ACC) of timer number *6* within data file *T4*.

file i03604

Question 4

In order to learn PLC programming and perform the exercises necessary for exams in this course, you must have your own PLC *trainer* consisting of a working PLC and input switches all wired and ready to use.



All components should be securely mounted to a wood board or some other structure making it easy to transport and use. You *must* have a terminal block in between the switches, indicators, and PLC I/O terminals to allow for easy connection and disconnection of external devices to your PLC without wearing out the screws on the PLC's terminal block prematurely. Separate terminal blocks are easily replaced, whereas the terminal block on your PLC is likely much more expensive and inconvenient to replace!

Consult the user's manual for your PLC in order to determine how all devices should be wired to the input and output (I/O) terminals. Note that often there are different types of I/O (AC, DC, sourcing, sinking) available for the same (or similar) model of PLC. Most PLC user's manuals give detailed diagrams showing how to connect devices to discrete I/O points, so be sure to follow the proper diagram for your specific PLC model!

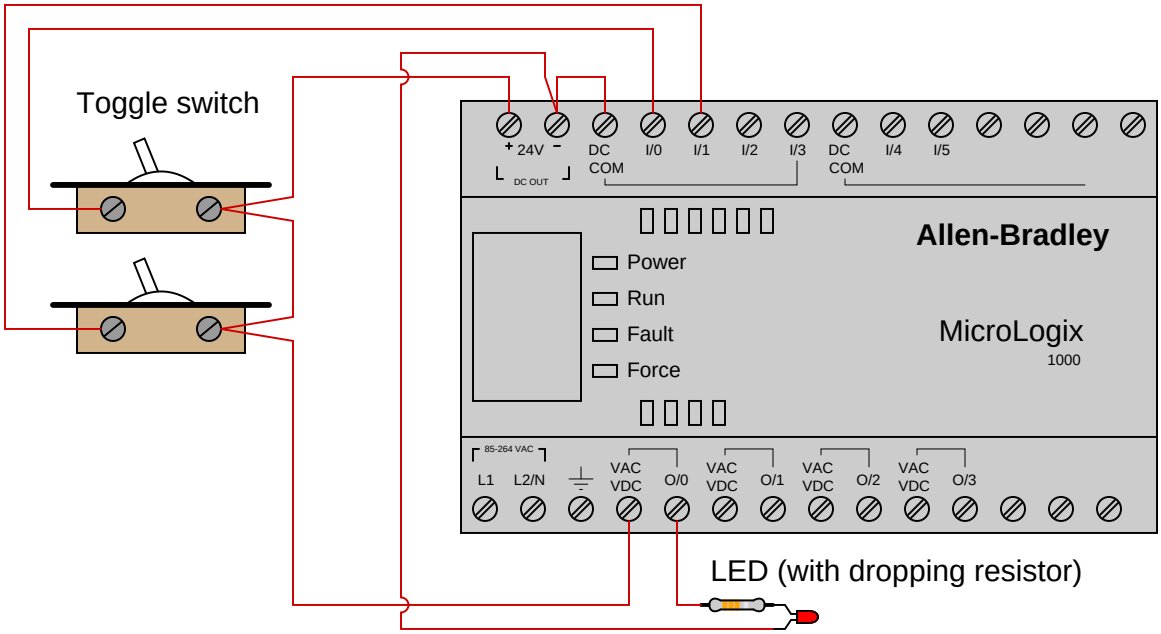
Once you have your PLC wired, the next step is to install and run the software used to program your programmable logic controller (PLC), and try to get the two devices communicating with each other. This, of course, requires you have a special cable connecting your PC to your PLC, with any necessary "drivers" installed on your PC to allow it to communicate. Like all serial-based communications, the PC needs to be properly configured with regard to bit rate, number of data bits, number of stop bits, and parity in order to communicate with the PLC. The software you will be using should have an "auto detect" feature which will sequentially try various combinations of these parameters until it finds one combination that works. Note: on Allen-Bradley PLCs, you must first install and run software called *RSLinx* which manages communications between your PC and PLC, before you start up the programming software (*RSLogix*).

After that, your next step is to use programming software (installed in a personal computer) to program your PLC with some simple function consisting of "contact" and "coil" instructions. The purpose of a virtual contact in a PLC program is to *read* data bits from memory, while the purpose of a virtual coil in a PLC program is to *write* data bits to memory. Thus, you will create programs for the PLC using virtual contacts to read the states of real-world switches connected to inputs on the PLC, and using virtual coils to control real-world outputs on the PLC to energize loads such as lamps and solenoids. The interconnections and arrangements of these virtual contacts and coils determine the logic implemented by the PLC: specifying the conditions necessary to energize real-world devices based on input conditions.

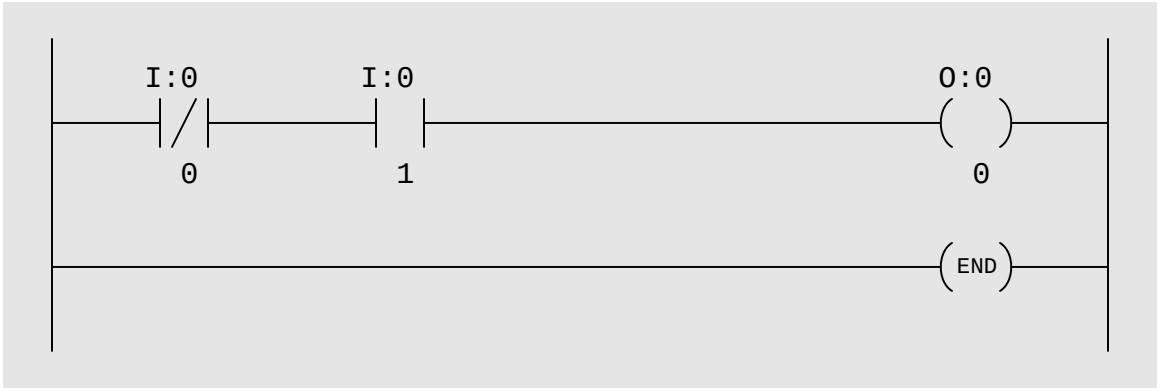
You will find step-by-step instructional tutorials for both Allen-Bradley MicroLogix and Koyo CLICK PLCs in your Instrumentation Reference (provided by the instructor). Follow these tutorials to establish communication between your PC and your PLC, and to write a simple contact-and-coil ladder diagram program, before attempting the exercises that follow. You will also find much pertinent information for programming Allen-Bradley MicroLogix PLCs in the *RSLogix 500 Getting Results Guide*, since the SLC 500

line of Allen-Bradley PLCs program so similarly to the MicroLogix line.

This example shows an Allen-Bradley MicroLogix 1000 series PLC (model 1761-L10BWA) wired to two toggle switches and one LED indicator lamp, complete with a demonstration program. Note that line power (120 VAC) wire connections to power the PLC have been omitted, so the focus is solely on the I/O wiring:



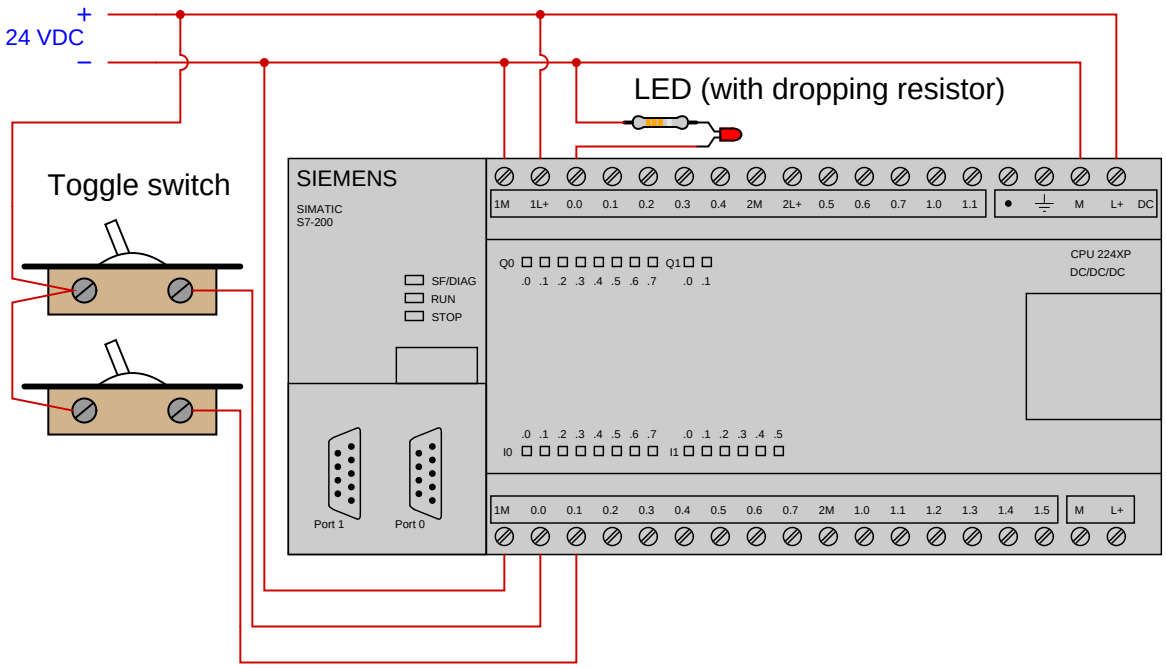
Ladder-Diagram program written to PLC:



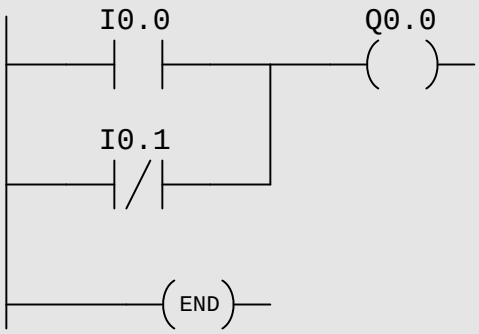
Note how Allen-Bradley I/O is labeled in the program: input bits designated by the letter `I` and output bits designated by the letter `O`.

Based on the wiring and program you see for this PLC, identify the switch state combinations resulting in an energized lamp. Try duplicating this program in your own PLC (even if it is a different brand or model) and see how it functions. Be sure to activate the *color highlighting* feature of your programming editor so you may see the "live" status of the program's virtual contacts and coil!

This example shows a Siemens S7-200 series PLC (model 224XP) wired to two toggle switches and one LED indicator lamp, complete with a demonstration program:


+
24 VDC
–
LED (with dropping resistor)
Toggle switch
SIEMENS
SIMATIC
S7-200
SF/DIAG
RUN
STOP
Port 1
Port 0
CPU 224XP
DC/DC/DC
1M 1L+ 0.0 0.1 0.2 0.3 0.4 2M 2L+ 0.5 0.6 0.7 1.0 1.1
M L+ DC
Q0 .0 .1 .2 .3 .4 .5 .6 .7 Q1 .0 .1
I0 .0 .1 .2 .3 .4 .5 .6 .7 I1 .0 .1 .2 .3 .4 .5
1M 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 2M 1.0 1.1 1.2 1.3 1.4 1.5
M L+

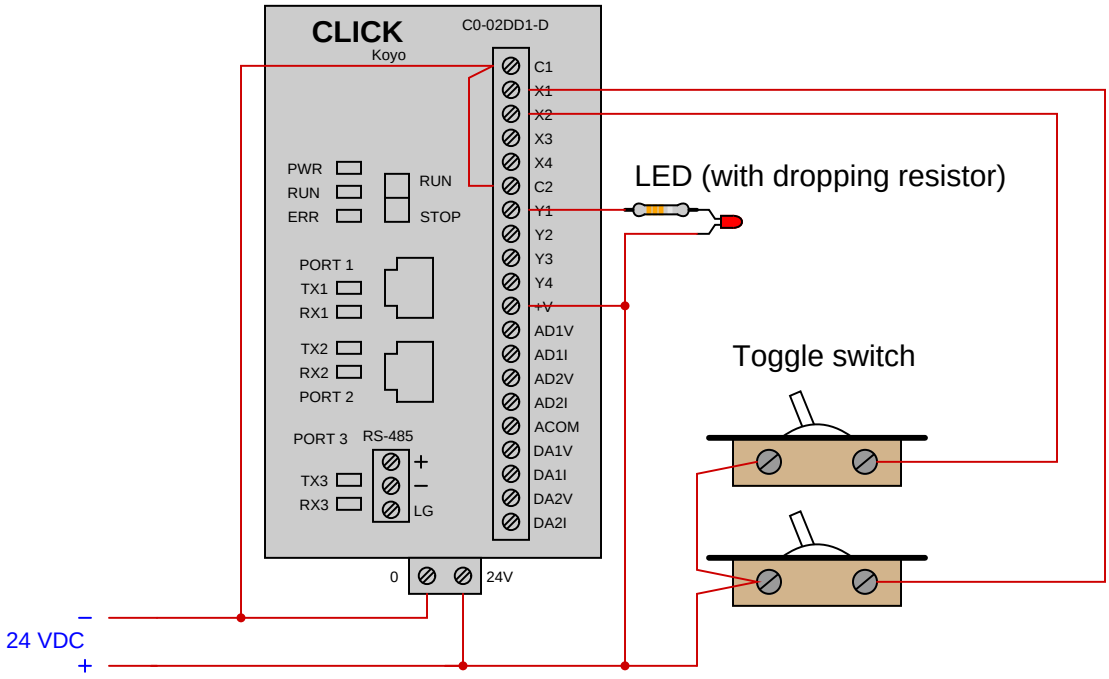
Ladder-Diagram program written to PLC:



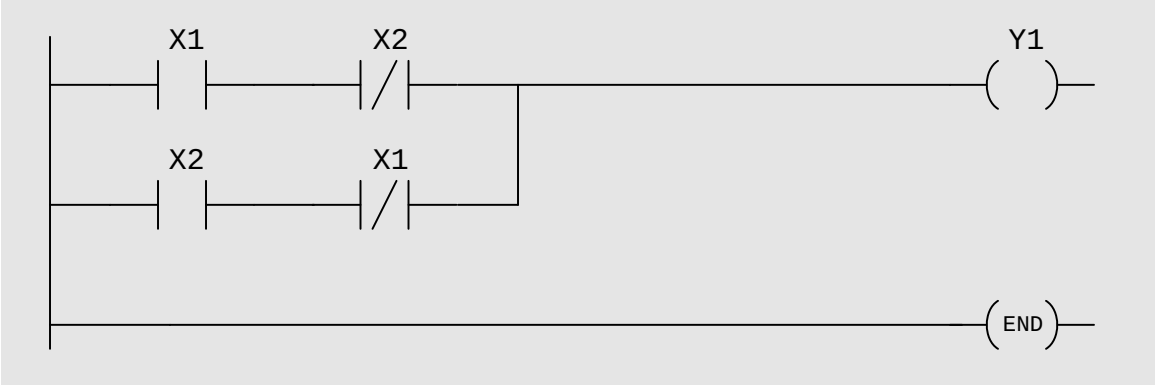
Note how Siemens I/O is labeled in the program: input bits designated by the letter `I` and output bits designated by the letter `Q`.

Based on the wiring and program you see for this PLC, identify the switch state combinations resulting in an energized lamp. Try duplicating this program in your own PLC (even if it is a different brand or model) and see how it functions. Be sure to activate the *color highlighting* feature of your programming editor so you may see the "live" status of the program's virtual contacts and coil!

This example shows a Koyo "CLICK" PLC (model C0-02DD1-D) wired to two toggle switches and one LED indicator lamp, complete with a demonstration program:



Ladder-Diagram program written to PLC:



Note how Koyo I/O is labeled in the program: input bits designated by the letter `X` and output bits designated by the letter `Y`.

Based on the wiring and program you see for this PLC, identify the switch state combinations resulting in an energized lamp. Try duplicating this program in your own PLC (even if it is a different brand or model) and see how it functions. Be sure to activate the *color highlighting* feature of your programming editor so you may see the "live" status of the program's virtual contacts and coil!

file i04513

Question 5

The most basic type of real-world input to a PLC is a *discrete* (on/off) input. Each discrete input channel on a PLC is associated with a single bit in the PLC's memory. Use the PLC programming software on your personal computer to "connect" to your PLC, then locate the facility within this software that allows you to monitor the status of your PLC's discrete input bits.

Actuate the switches connected to your PLC's discrete input channels while watching the status of the respective bits. Based on what you see, what does a "1" bit status signify, and what does a "0" bit status signify?

Suggestions for Socratic discussion

- How does your PLC address discrete input bits? In other words, what is the convention it uses to label these bits, and distinguish them from each other?
- How does the programming software for your PLC provide access to discrete input bit status?

PLC comparison:

- Allen-Bradley Logix 5000: the *Controller Tags* folder (typically on the left-hand pane of the programming window set) lists all the tag names defined for the PLC project, allowing you to view the real-time status of them all. Discrete inputs do not have specific input channel tag names, as tag names are user-defined in the Logix5000 PLC series.
- Allen-Bradley PLC-5, SLC 500, and MicroLogix: the *Data Files* listing (typically on the left-hand pane of the programming window set) lists all the data files within that PLC's memory. Opening a data file window allows you to view the real-time status of these data points. Discrete inputs are the `I` file addresses (e.g. `I:0/2`, `I:3/5`, etc.). The letter "`I`" represents "input," the first number represents the slot in which the input card is plugged, and the last number represents the bit within that data element (a 16-bit word) corresponding to the input card.
- Siemens S7-200: the *Status Chart* window allows the user to custom-configure a table showing the real-time values of multiple addresses within the PLC's memory. Discrete inputs are the `I` memory addresses (e.g. `I0.1`, `I1.5`, etc.).
- Koyo (Automation Direct) DirectLogic and CLICK: the *Data View* window allows the user to custom-configure a table showing the real-time values of multiple addresses within the PLC's memory. Discrete inputs are the `X` memory addresses (e.g. `X1`, `X2`, etc.).

file i01876

Question 6

The most basic type of real-world output from a PLC is a *discrete* (on/off) output. Each discrete output channel on a PLC is associated with a single bit in the PLC's memory. Use the PLC programming software on your personal computer to "connect" to your PLC, then locate the facility within this software that allows you to monitor the status of your PLC's discrete output bits.

Use the "force" utility in the programming software to force different output bits to a "1" status. Based on what you see, what does a "1" bit status signify, and what does a "0" bit status signify?

Is there any visual indication that bits have been forced from their normal state(s) in your PLC? Note that "forcing" causes the PLC to output the values you specify, whether or not the programming in the PLC "wants" those bits to have those forced values!

Suggestions for Socratic discussion

- How does your PLC address discrete output bits? In other words, what is the convention it uses to label these bits, and distinguish them from each other?
- How does the programming software for your PLC provide access to discrete output bit status, and the ability to force them?
- Why would anyone ever wish to force an output bit in a PLC, especially if doing so overrides the logic programmed into the PLC?

PLC comparison:

- Allen-Bradley Logix 5000: forces may be applied to specific tag names by right-clicking on the tag (in the program listing) and selecting the "Monitor" option. Discrete outputs do not have specific output channel tag names, as tag names are user-defined in the Logix5000 PLC series.
- Allen-Bradley PLC-5, SLC 500, and MicroLogix: the *Force Files* listing (typically on the left-hand pane of the programming window set) lists those data files within the PLC's memory liable to forcing by the user. Opening a force file window allows you to view and set the real-time status of these bits. Discrete outputs are the `O` file addresses (e.g. `O:0/7`, `O:6/2`, etc.). The letter "`O`" represents "output," the first number represents the slot in which the output card is plugged, and the last number represents the bit within that data element (a 16-bit word) corresponding to the output card.
- Siemens S7-200: the *Status Chart* window allows the user to custom-configure a table showing the real-time values of multiple addresses within the PLC's memory, and enabling the user to force the values of those addresses. Discrete outputs are the `Q` memory addresses (e.g. `Q0.4`, `Q1.2`, etc.).
- Koyo (Automation Direct) DirectLogic and CLICK: the *Override View* window allows the user to force variables within the PLC's memory. Discrete outputs are the `Y` memory addresses (e.g. `Y1`, `Y2`, etc.).

file i01877

Question 7

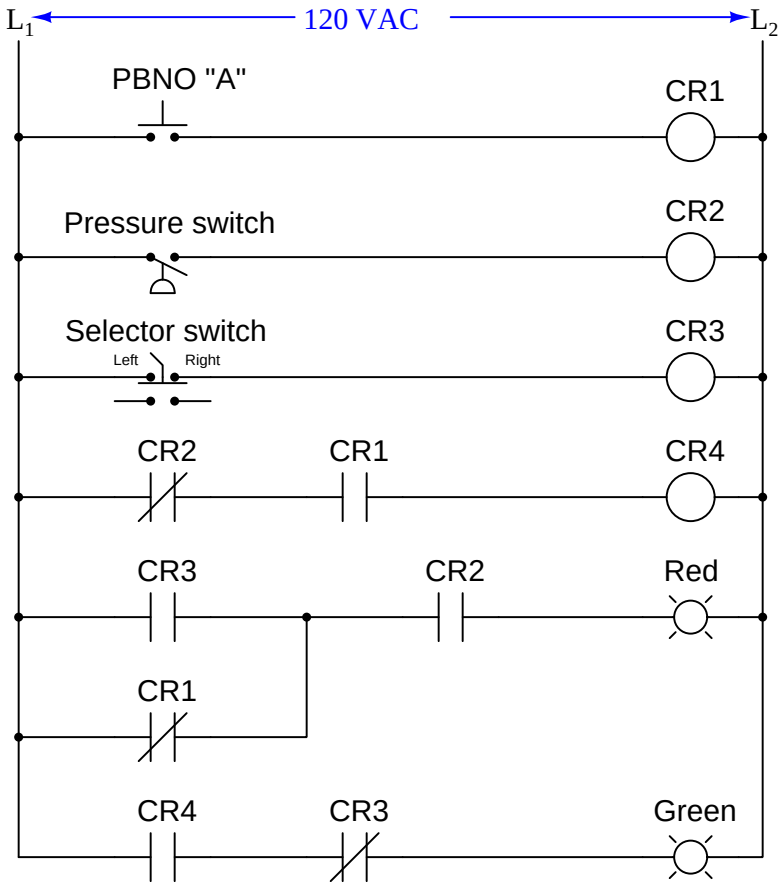
Read and outline the "Relating I/O Status to Virtual Elements" subsection of the "Logic Programming" section of the "Programmable Logic Controllers" chapter in your *Lessons In Industrial Instrumentation* textbook. Note the page numbers where important illustrations, photographs, equations, tables, and other relevant details are found. Prepare to thoughtfully discuss with your instructor and classmates the concepts and examples explored in this reading.

The fundamental concept of relating I/O status to program elements is not necessarily easy to grasp. For this reason, a "Process Switches and PLC Circuits" worksheet has been placed in the *Socratic Instrumentation* practice worksheet collection. Feel free to use this practice worksheet to supplement your studies on this critically important topic!

file i04516

Question 8

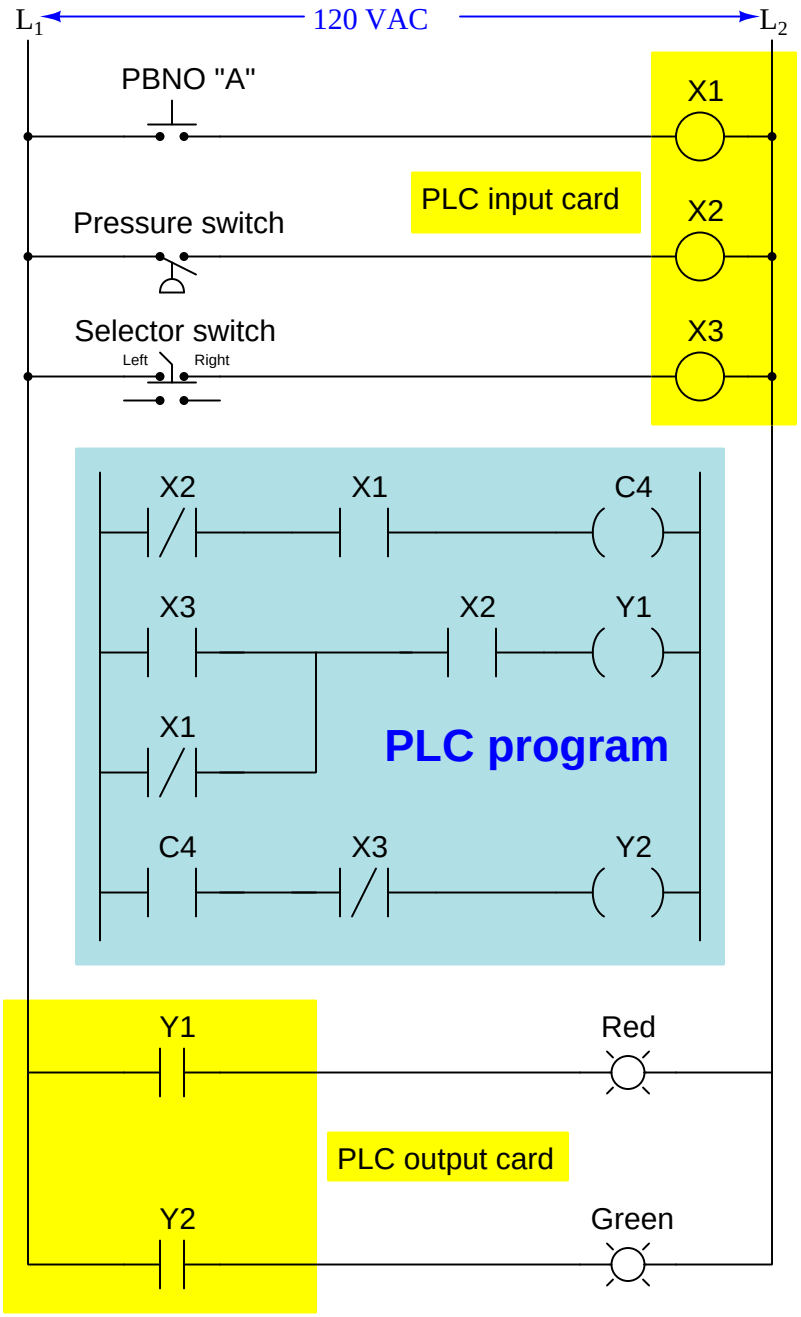
Analyze the status of all relay contacts and lamps in this hard-wired relay "ladder logic" control circuit:



Assume the following input conditions:

- Pushbutton switch *unpressed*
- Pressure *above* trip threshold
- Selector switch in its *right-hand* position

Now, analyze the status of this PLC-controlled system assuming the same input conditions. Note the distinction between the 120 VAC circuitry and the "virtual circuit" in the blue-shaded area representing the program executed by the PLC's microprocessor:



- Pushbutton switch *unpressed*
- Pressure *above* trip threshold
- Selector switch in its *right-hand* position

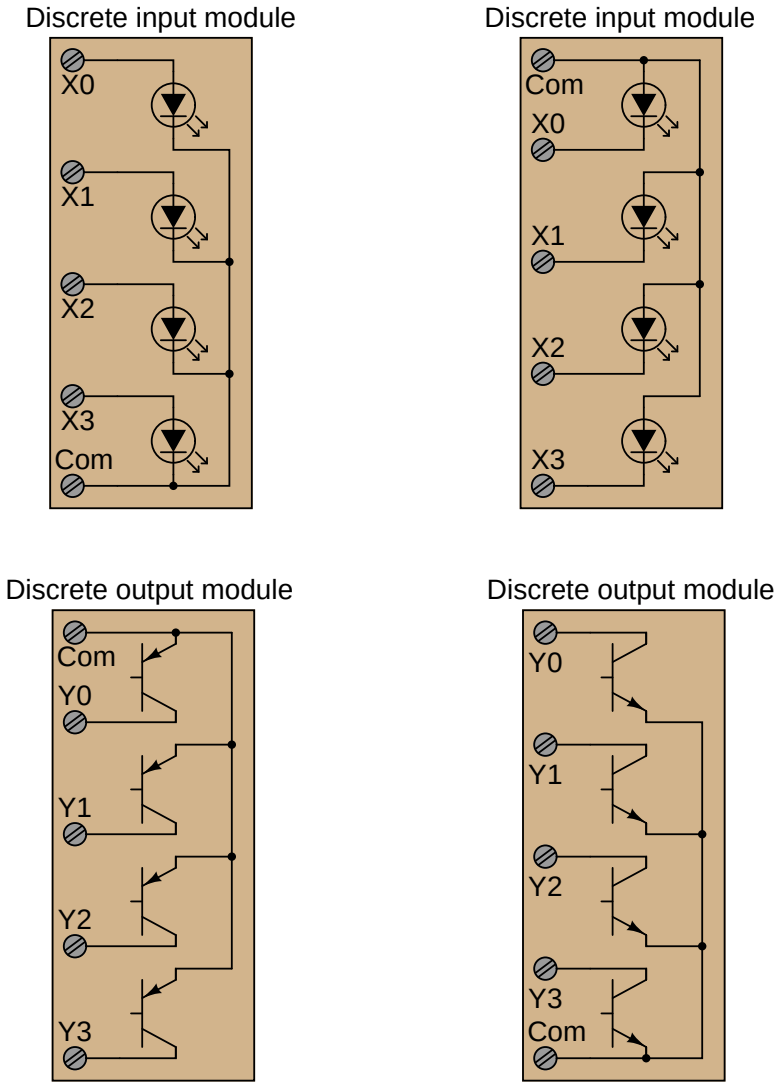
How is the PLC-controlled system similar to the hard-wired relay control system? How is it different?

file i02605

Question 9

Discrete (on/off) I/O for PLCs often works on AC (alternating current) power. AC input circuitry usually consists of an optocoupler (LED) with rectification and a large dropping resistor to allow 120 volt AC operation. AC output circuitry usually consists of TRIACs. Explain how both of these technologies work.

DC I/O for a PLC generally consists of optocoupled LEDs for inputs and bipolar transistors for outputs. Some examples are shown in the following schematics. Note carefully the different variations:



Determine for each of these input and output module types, whether they would be properly designated *sourcing* or *sinking*.

Suggestions for Socratic discussion

- Determine how real input and output devices (e.g. switches, solenoid coils) would need to be connected to the I/O terminals of these modules.

file i02359

Question 10

Have some fun writing simple "exploratory" or "demonstration" ladder-diagram PLC programs to perform different functions. Feel free to explore the following instruction types:

- Contacts and coils
- Counters (up, down, up/down)
- Timers (on-delay, off-delay, retentive)
- Sequencing instructions
- Math instructions (add, subtract, multiply, divide)

Identify some realistic applications for PLC programs using counters and timers. What sorts of real-life processes might benefit from a PLC function where something turns on or off after a definite number of counts applied to the PLC input, or after a certain amount of time has passed?

Note: this simple exercise may seem trivial, but it holds the key to self-instruction on PLC programming! Having your very own PLC to work with in the classroom is a tremendously powerful learning tool. Whenever you encounter a new programming instruction (e.g. a timer, a math instruction, etc.) that you do not yet know how to use, you may explore that instruction's properties and behavior by creating a simple program in your PLC with nothing but that instruction. Your PLC's *User Manual* or *Instruction Set* reference manual will show you the basic syntax of the instruction, which you may copy verbatim as an example. Once this simple program is loaded into your PLC's memory, you can "play" with it to see its live behavior while viewing the program online.

Once you have directly observed how the instruction works, your next step is to add comments to the program describing how that instruction works in your own words. Be as detailed as possible here, treating this activity as though you were asked to explain everything to someone who knew absolutely nothing about the instruction. These comments will serve as notes to yourself later, when you need to refresh your memory on how a particular instruction functions or what it is used for.

Do not be surprised if your instructor asks you to show your demonstration program(s) for particular instructions in the future! If you experience difficulty using a particular instruction in a programming assignment, your instructor may check to see if you have created and run a demonstration program to learn how that instruction is supposed to function.

Refer to the "Answer" section of this question to see some examples of what such a demonstration program might look like.

Suggestions for Socratic discussion

- A helpful tip when writing your own demonstration programs is to save each one with a filename that makes it easy to locate on your personal computer. For example, you might wish to name each of your demonstration programs beginning with the word "Demo" and using underscore characters to separate descriptive words (or instruction names) in the rest of the filename. Some examples are shown here:
  → `Demo_contacts_coils`
  → `Demo_upcounter`
  → `Demo_downcounter`
  → `Demo_TOF_timer`
  → `Demo_TON_timer`
  → `Demo_ADD_instruction`

  file i00120

Question 11

All PLCs provide "special" locations in memory holding values useful to the programmer, such as status warnings, real-time clock settings, calendar dates, etc. Use the PLC programming software on your personal computer to "connect" to your PLC, then locate the facility within this software that allows you to explore some of these locations in memory.

Identify some of the specific status-related and "special" memory locations in your PLC, and comment on those you think might be useful to use in the future. Note the following memory types you may find associated with these addresses:

- Boolean (discrete) = simply on or off (1 or 0)
- Integer = whole-numbered values
- Floating-point ("real") = fractional values

Suggestions for Socratic discussion

- Describe some of the "special" memory locations you find in your search, and comment on how some of them might be useful.
- One of the useful bits provided by many PLCs is a "flashing" bit that simply turns on and off at regular intervals. How many of these bits can you find in your PLC's memory, and how rapidly does each one oscillate?

PLC comparison:

- Allen-Bradley Logix 5000: various "system" values are accessed via `GSV` (Get System Value) and `SSV` (Save System Value) instructions.
- Allen-Bradley PLC-5, SLC 500, and MicroLogix: the *Data Files* listing (typically on the left-hand pane of the programming window set) shows file number 2 as the "Status" file, in which you will find various system-related bits and registers.
- Siemens S7-200: the *Special Memory* registers contain various system-related bits and registers. These are the `SM` memory addresses (e.g. `SM0.1`, `SMB8`, `SMW22`, etc.).
- Koyo (Automation Direct) DirectLogic and CLICK: the *Special* registers contain various system-related bits and registers. These are the `SP` memory addresses (e.g. `SP1`, `SP2`, `SP3`, etc.) in the DirectLogic PLC series, and the `SC` and `SD` memory addresses in the CLICK PLC series.

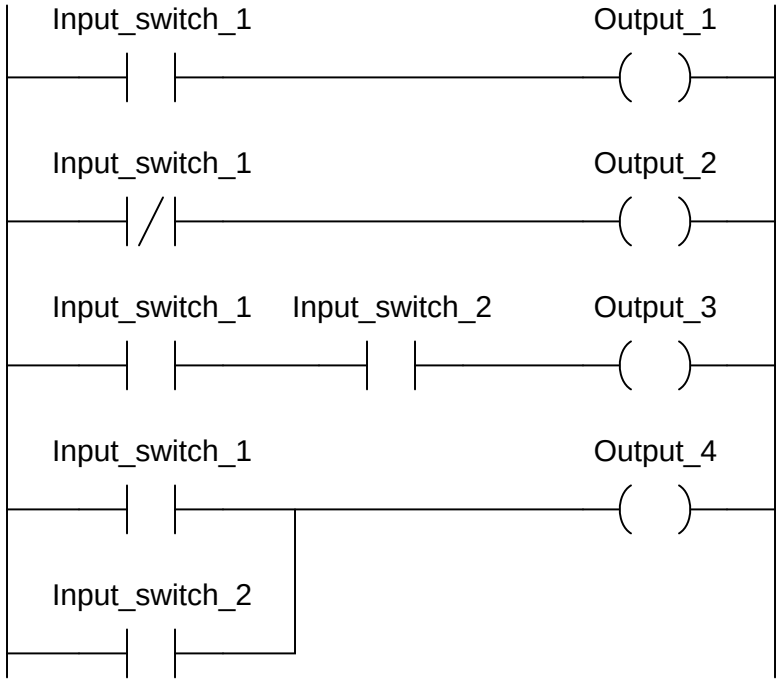
file i01878

Question 12

Write a PLC program that accepts two discrete input signals (from two switches), and outputs the following four discrete outputs:

- Output channel #1: The status of input switch #1 (simply repeating input #1)
- Output channel #2: The Boolean complement (opposite) of input switch #1
- Output channel #3: The `AND` function of switches #1 and #2
- Output channel #4: The `OR` function of switches #1 and #2

Shown here is a generic RLL listing of such a program:



Turn on status highlighting within the programming software environment so that you may see the virtual "power" flow through the "conductive" contacts as you test the program.

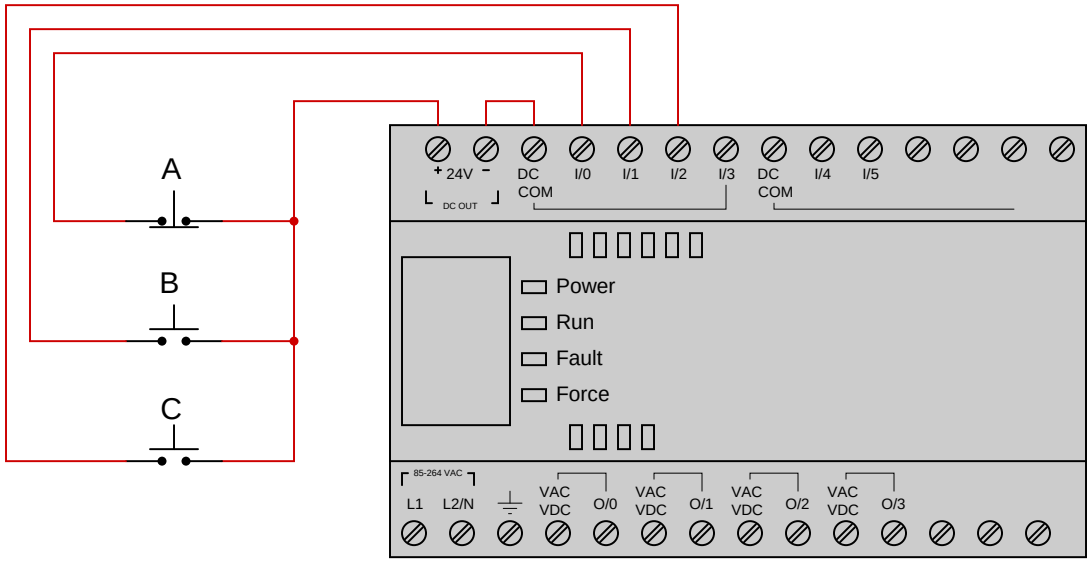
Suggestions for Socratic discussion

- How are discrete input and output points associated with contacts and coils in the ladder-logic program?
- How do you draw vertical connecting lines in the ladder-logic program?
- How do you assign "alias" names to inputs and outputs for easier program readability? For example, how do you assign an English name to the input `I:2/4` (Input channel 4 on card 2) on an Allen-Bradley SLC 500 PLC so that it reads as "Input_switch_4" in the program instead of "`I:2/4`" in the programming software's display?
- Where is the software function (pull-down menu option, button, hot-key, etc.) located that allows you to turn on contact status highlighting in the PLC programming software?

file i03667

Question 13

Suppose we have an Allen-Bradley MicroLogix 1000 PLC connected to three momentary-contact pushbutton switches as shown in this illustration:

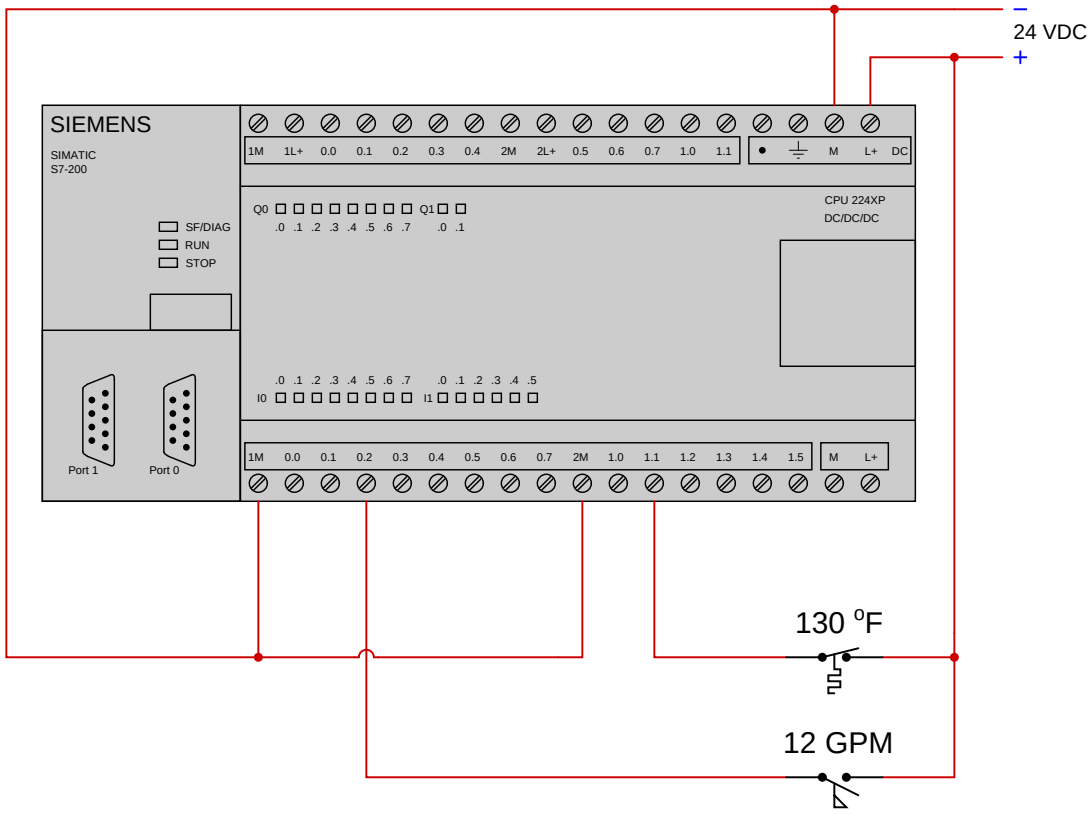


Determine the bit statuses of `I:0/0`, `I:0/1`, and `I:0/2` when switch A is unpressed (released), switch B is unpressed (released), and switch C is pressed.

file i01865

Question 14

Suppose we have a Siemens S7-200 PLC connected to two process switches as shown in this illustration:

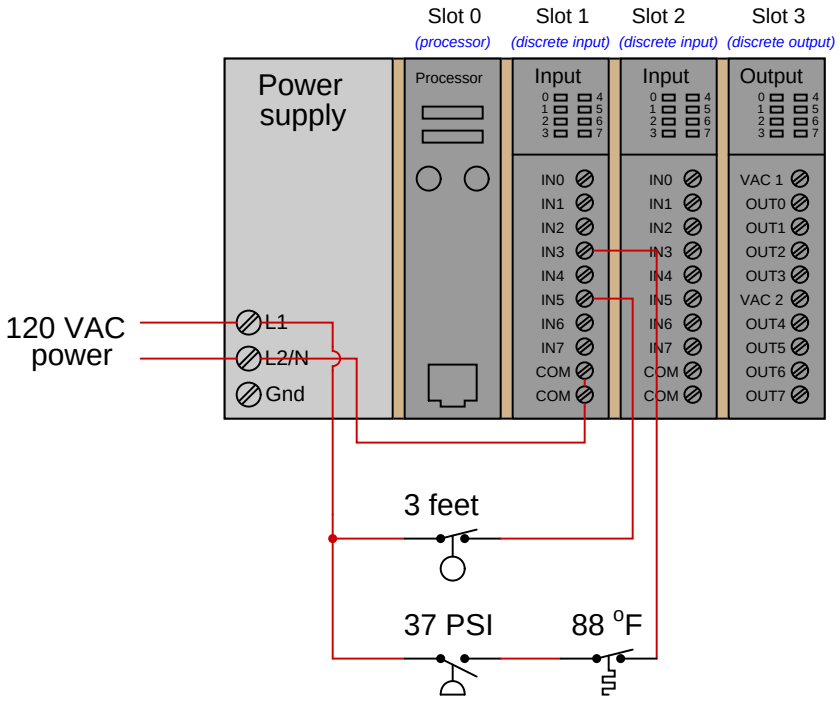


Determine the bit statuses of `I0.2` and `I1.1` when the temperature switch senses 194 ^{o}F and the flow switches senses 19 GPM.

file i01871

Question 15

Suppose we have an Allen-Bradley SLC 500 PLC connected to two process switches as shown in this illustration:



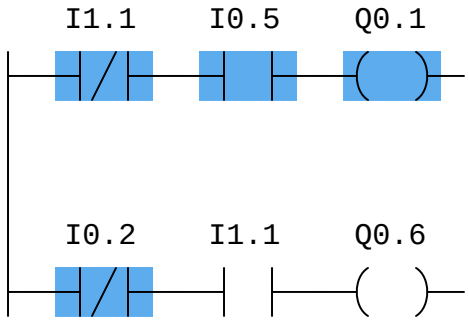
Determine the process conditions necessary to generate the following input bit statuses in the PLC's memory:

- `I:1/3` = 1
- `I:1/5` = 0

file i01872

Question 16

Examine this "live" display of a Siemens S7-300 PLC's program, and from this determine all bit statuses represented by the color highlighting in this ladder logic program:

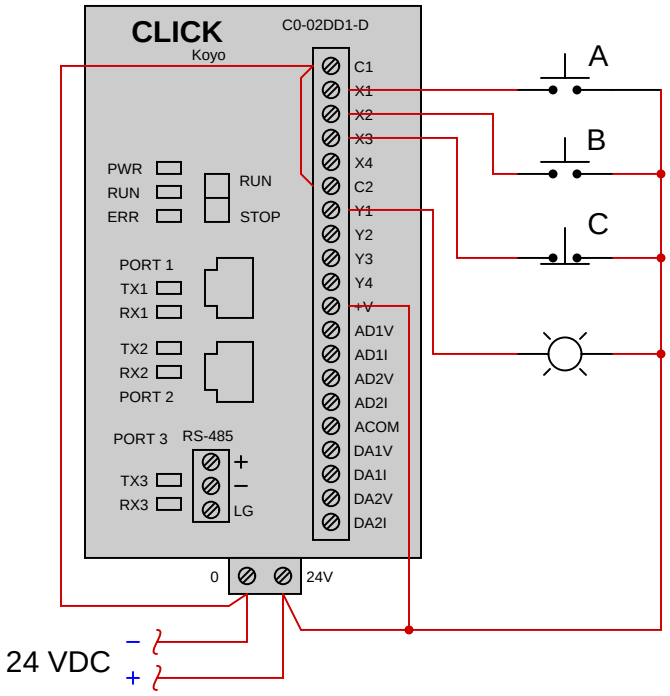


- I0.2 = ???
- I0.5 = ???
- I1.1 = ???
- Q0.1 = ???
- Q0.6 = ???

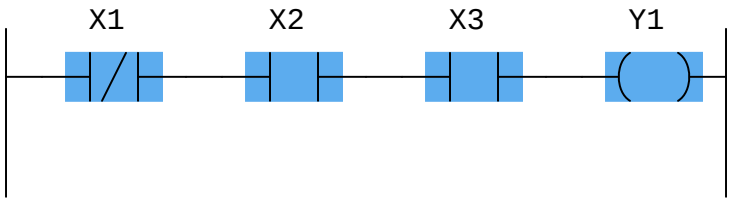
file i01873

Question 17

Suppose we have a Koyo "CLICK" PLC connected to three momentary-contact pushbutton switches as shown in this illustration:



Determine the switch actuation statuses (i.e. pressed versus released) given the "live" display of the ladder logic program shown here:

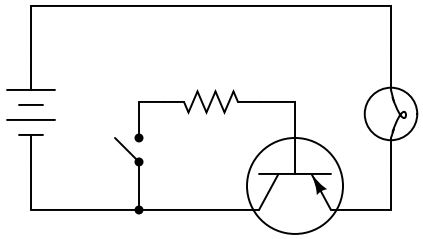


Also, determine the status of the lamp connected to the PLC's `Y1` output.

file i01874

Question 18

Explain the function of this light-switching circuit, tracing the directions of all currents when the switch closes:

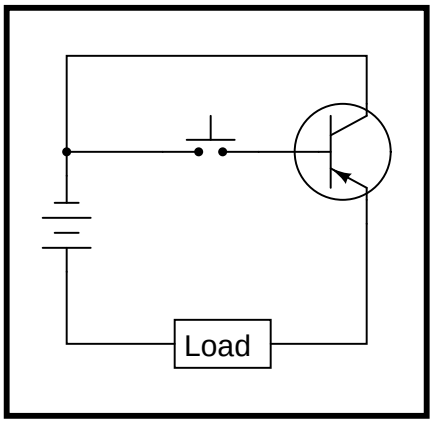


file i01000

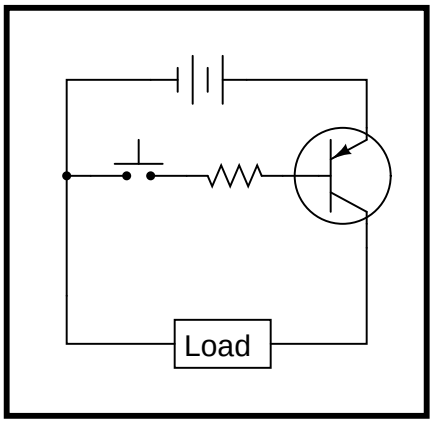
Question 19

Some of the following transistor switch circuits are properly configured, and some are not. Identify which of these circuits will function properly (i.e. turn on the load when the switch closes) and which of these circuits are mis-wired:

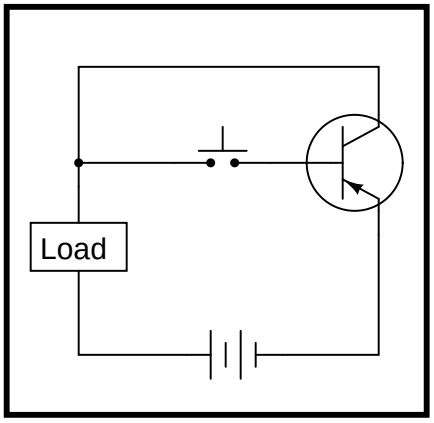
Circuit 1



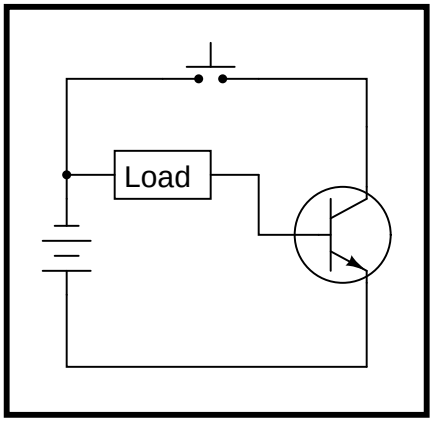
Circuit 2



Circuit 3



Circuit 4

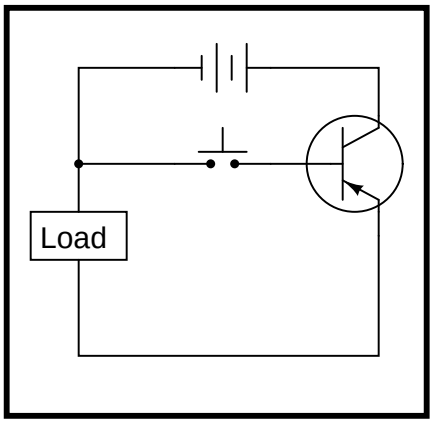


file i01002

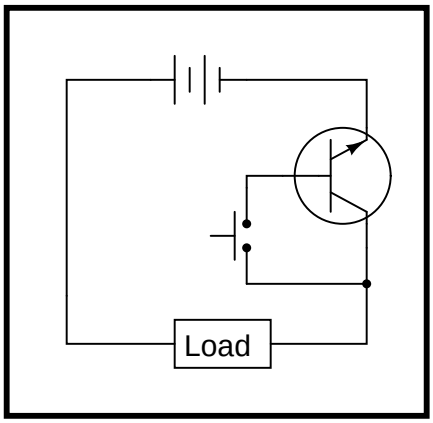
Question 20

Some of the following transistor switch circuits are properly configured, and some are not. Identify which of these circuits will function properly (i.e. turn on the load when the switch closes) and which of these circuits are mis-wired:

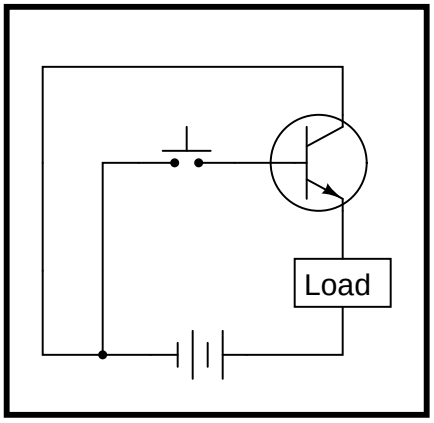
Circuit 1



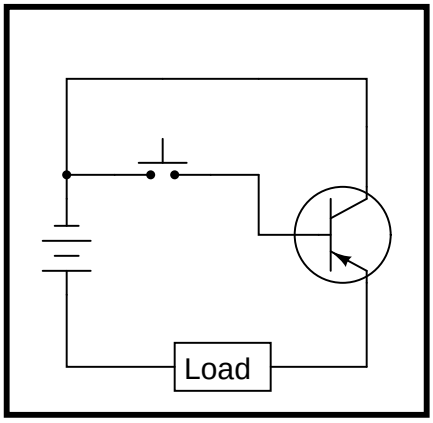
Circuit 2



Circuit 3



Circuit 4



file i01003

Question 21

Read and outline the "Contacts and Coils" subsection of the "Ladder Diagram (LD) Programming" section of the "Programmable Logic Controllers" chapter in your *Lessons In Industrial Instrumentation* textbook. Note the page numbers where important illustrations, photographs, equations, tables, and other relevant details are found. Prepare to thoughtfully discuss with your instructor and classmates the concepts and examples explored in this reading.

Suggestions for Socratic discussion

- If you have access to your own PLC for experimentation, I urge you to write a simple *demonstration* program in your PLC allowing you to explore the behavior of these PLC instructions. The program doesn't have to do anything useful, but merely demonstrate what each instruction does. First, read the appropriate section in your PLC's manual or instruction reference to identify the proper syntax for that instruction (e.g. which types of data it uses, what address ranges are appropriate), then write the simplest program you can think of to demonstrate that function in isolation. Download this program to your PLC, then run it and observe how it functions "live" by noting the color highlighting in your editing program's display and/or the numerical values manipulated by each instruction. After "playing" with your demonstration program and observing its behavior, write comments for each rung of your program explaining in your own words what each instruction does.

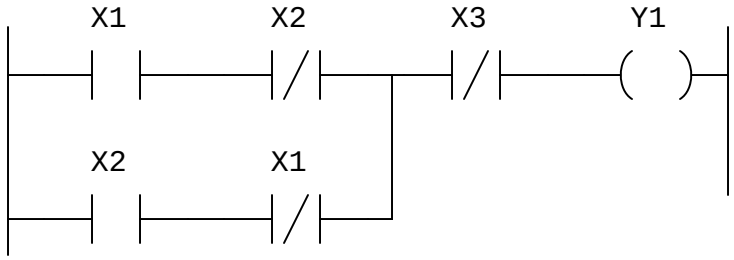
  file i04517

Question 22

Suppose a Siemens 545 PLC has the following input bit states:

- X1 = 0
- X2 = 1
- X3 = 0

Sketch color highlighting for the contacts and coils in the PLC's program given these bit statuses, also determining the status of output bit Y1:



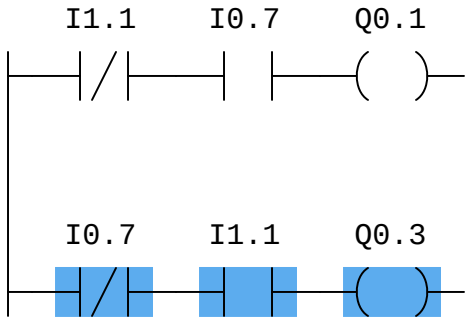
Suggestions for Socratic discussion

- PLC training expert Ron Beaufort teaches students to think of a "normally-open" PLC program contact instruction as a command to the PLC's processor to *"Go look for a 1"*. Conversely, he teaches students to think of a "normally-closed" instruction as a command to *"Go look for a 0"*. Explain what Mr. Beaufort means by these phrases, and how this wisdom relates to this particular problem. Incidentally, Mr. Beaufort's excellent instructional videos (available freely on YouTube) are quite valuable to watch!
- Identify the significance of the labels "X" and "Y" for this PLC's bits. What do you suppose "X" signifies? What do you suppose "Y" signifies?
- Sketch a logic gate diagram implementing the same function as this PLC program.

file i04688

Question 23

Examine this "live" display of a Siemens S7-300 PLC's program, and from this determine all bit statuses represented by the color highlighting in this ladder logic program:



- I0.7 = ???
- I1.1 = ???
- Q0.1 = ???
- Q0.3 = ???

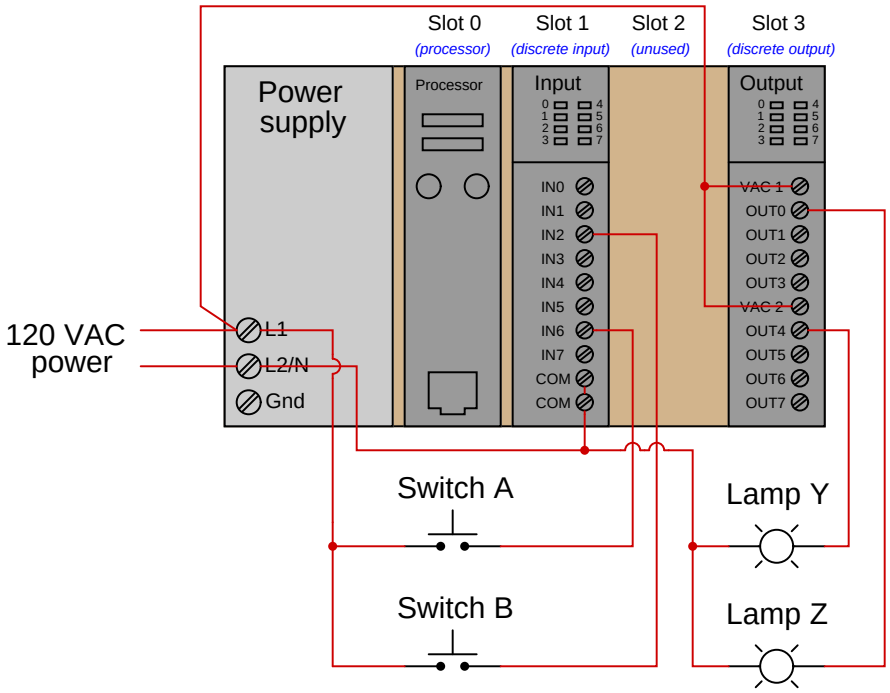
Suggestions for Socratic discussion

- PLC training expert Ron Beaufort teaches students to think of a "normally-open" PLC program contact instruction as a command to the PLC's processor to *"Go look for a 1"*. Conversely, he teaches students to think of a "normally-closed" instruction as a command to *"Go look for a 0"*. Explain what Mr. Beaufort means by these phrases, and how this wisdom relates to this particular problem. Incidentally, Mr. Beaufort's excellent instructional videos (available freely on YouTube) are quite valuable to watch!
- Identify the significance of the labels "I" and "Q" for this PLC's bits. What do you suppose "I" signifies? What do you suppose "Q" signifies?

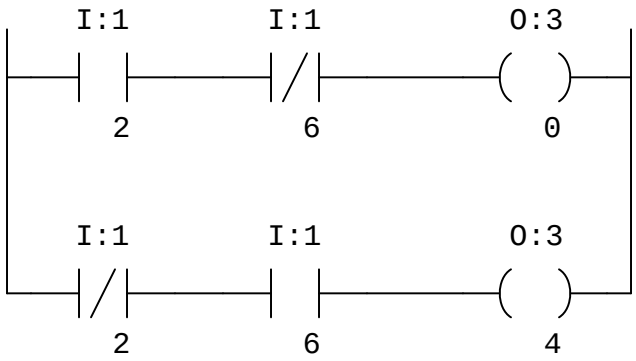
file i04689

Question 24

Suppose we have an Allen-Bradley model "SLC 500" PLC connected to a pair of momentary-contact pushbutton switches and light bulbs as shown in this illustration:



Examine the following relay ladder logic (RLL) program for this Allen-Bradley PLC, determining the statuses of the two lamps provided neither switch A nor switch B is pressed by a human operator:



Finally, draw color highlighting showing how these "contact" instructions will appear in an online editor program given the stated input conditions.

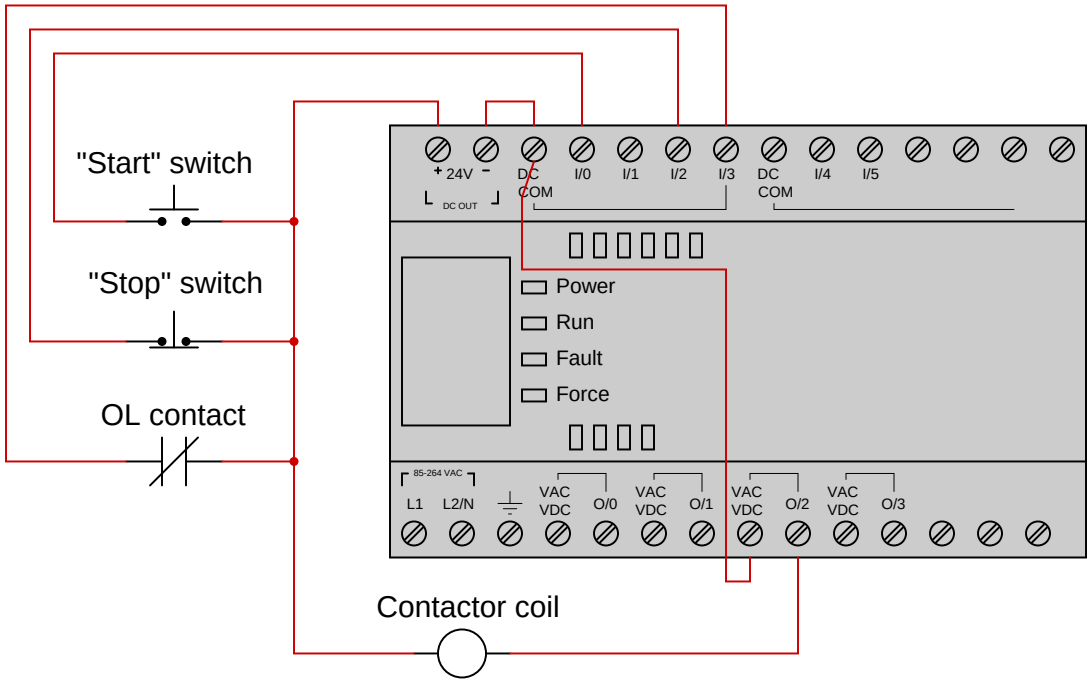
Suggestions for Socratic discussion

- Identify the significance of the labels "I" and "O" for this PLC's bits.
- Identify the significance of the first and second numbers in each bit label (e.g. the numbers "1" and "2" in the bit address `I:1/2`, for example). What pattern do you see as you compare the I/O connections with the respective contact instructions in the PLC program?

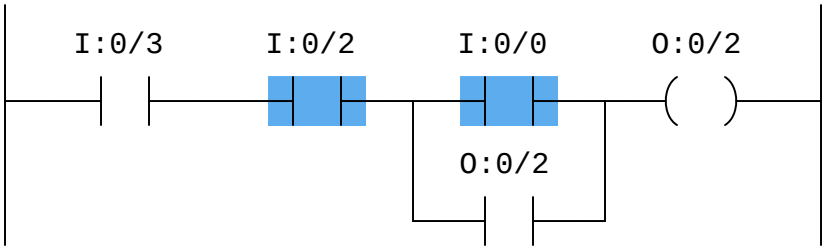
file i04628

Question 25

Suppose we have an Allen-Bradley MicroLogix 1000 controller connected to a pair of momentary-contact pushbutton switches and contactor controlling power to an electric motor as shown in this illustration:



This motor control system has a problem, though: the motor refuses to start when the "Start" pushbutton is pressed. Examine the "live" display of the ladder logic program inside this Allen-Bradley PLC to determine what the problem is, assuming an operator is continuously pressing the "Start" pushbutton as you examine the program:



Identify at least two causes that could account for all you see here.

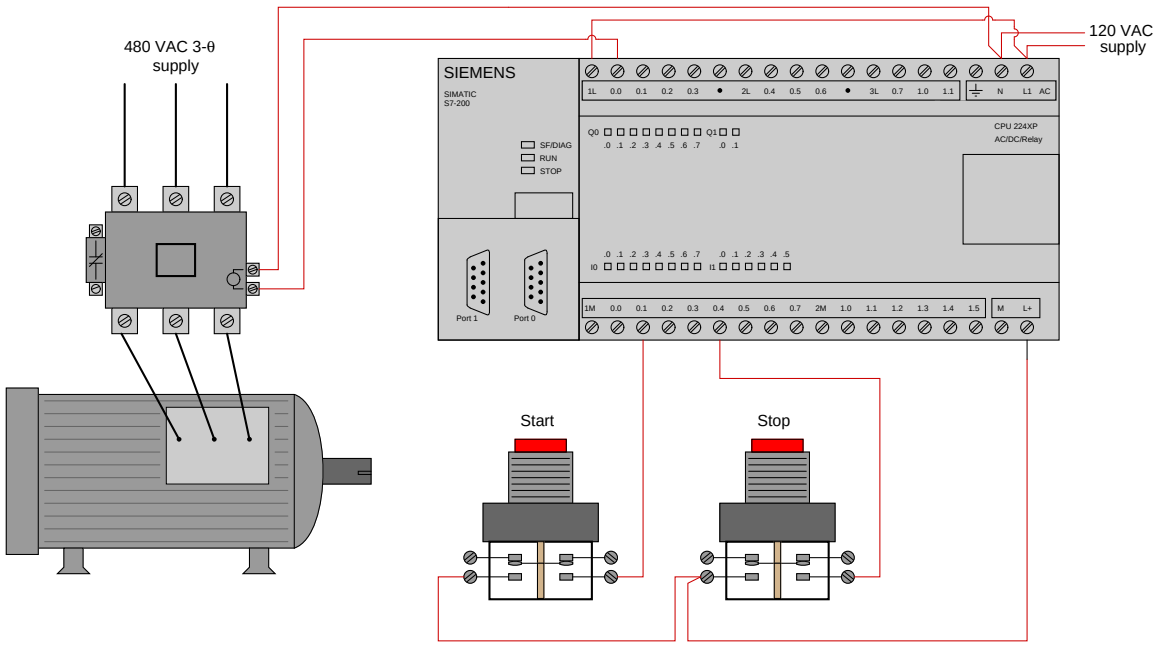
Suggestions for Socratic discussion

- Identify what your next troubleshooting step would be if you were tasked with solving this problem.
- A helpful problem-solving tip is to annotate each contact in the PLC program to show what its real-world function is. For example, contact `I:0/3` may be labeled "OL" because that is the real-world switch status it senses. Annotate all contacts in this program and explain how this annotation is helpful in analyzing the program.
- Describe the purpose of the contact labeled `O:0/2` in this program, explaining why it is often referred to as a *seal-in* contact.

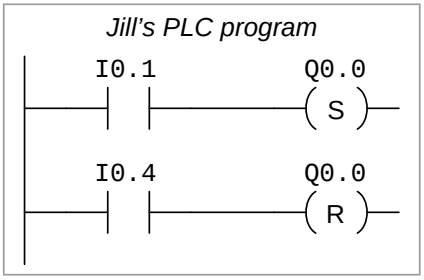
file i04662

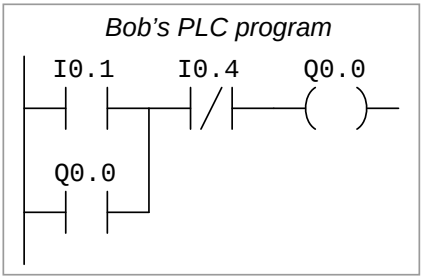
Question 26

Two technicians, Jill and Bob, work on programming Siemens S7-200 PLCs to control the starting and stopping of electric motors. Both PLCs are wired identically, as shown:



However, despite being wired identically, the two technicians' PLC programs are quite different. Jill's program uses *retentive coil* instructions ("Set" and "Reset" coils) while Bob's uses a "seal-in" contact instruction to perform the function of latching the motor on and off:





Explain how both of these PLC programs function properly to control the starting and stopping of the electric motor.

Suggestions for Socratic discussion

- It is ordinarily a bad thing to assign identical bit addresses to multiple coil instructions in a PLC program. With Jill's retentive coil program, however, this is not only permissible but in fact necessary for its proper operation. Explain why this is.
- A common misconception of students first learning PLC programming is to think that the type of contact instruction used in the PLC program must match the type of switch contact connected to that input (e.g. "A N.O. PLC instruction must go with a N.O. switch"). Explain why this is incorrect.
- Explain how both PLC programs will react if both the "start" and "stop" pushbuttons are simultaneously pressed.

- Alter both PLC programs to be "fail-safe" (i.e. shut the motor off) if ever the stop pushbutton switch fails circuit open.

file i03674

Question 27

Demonstration Program – contact and coil instructions

An important technique for learning any programming language – Ladder Diagram PLC programming included – is to write simple "demonstration" programs showcasing and explaining how particular instructions and programming constructs are supposed to work. Since you have access to your own personal PLC, you can explore the elements of your PLC's programming language like a scientist would explore new specimens: subject them to tests and record how they respond. This is how you will be able to teach yourself new models of PLC when you are working in your career, when you won't have textbooks to follow or training to show you exactly what to do.

Write such a "demonstration" program for your PLC's *contact and coil* instructions, where discrete inputs on your PLC control discrete outputs on your PLC. An acceptable demonstration program must meet these three criteria:

- **Simple** – nothing "extra" included in the program to detract from the fundamental behavior of the instruction(s) being explored
- **Complete** – nothing missing from the program relevant to the fundamental behavior of the instruction(s) being explored. *For a contact and coil demonstration program, this includes normally-open and normally-closed contact instructions, as well as regular and retentive coil instructions.*
- **Clearly documented** – every rung clearly commented in your own words, every variable named

Your instructor will challenge you to use this demonstration program to illustrate what you have learned about PLC counter instructions.

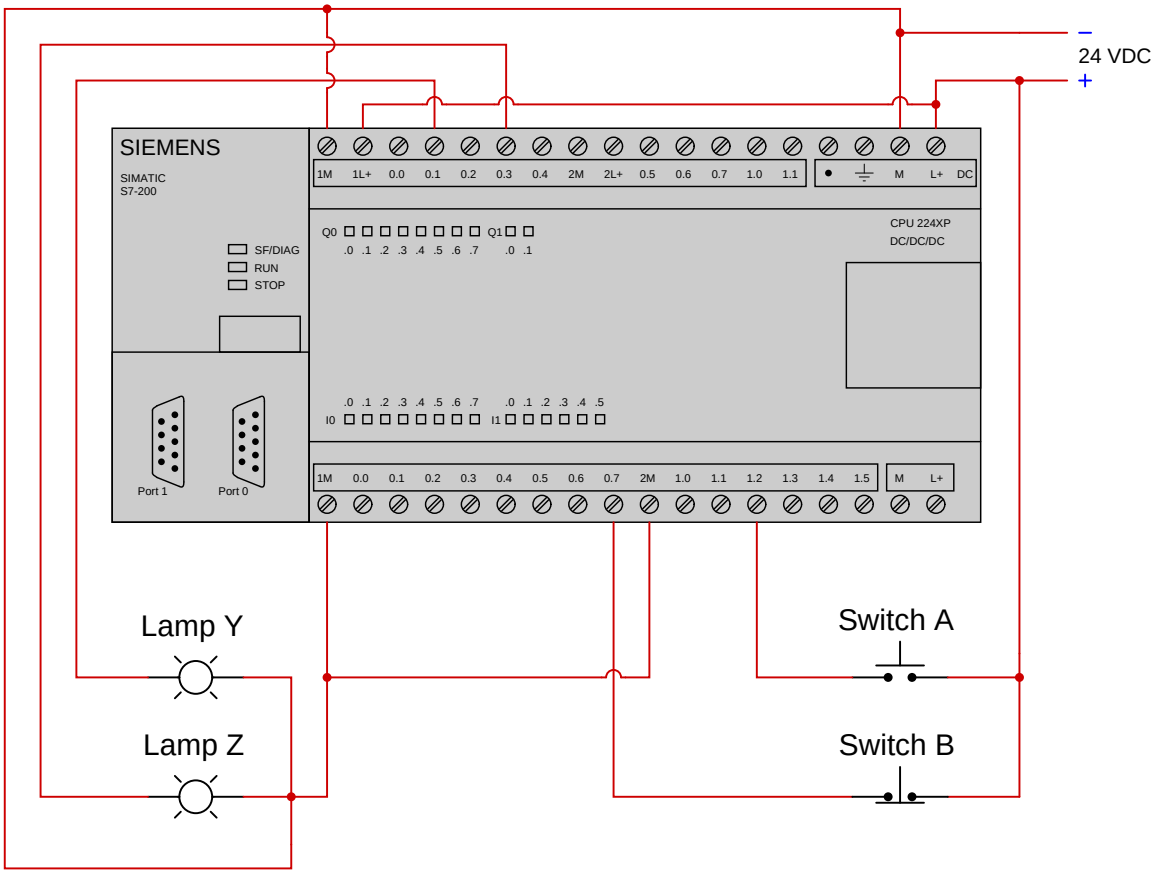
Suggested questions your demonstration program should answer:

- Identify the different contact instruction types offered on your PLC. Describe how each of them functions.
- Identify the different coil instruction types offered on your PLC. Describe how each of them functions.
- What happens when two contact instructions are linked to the same bit address in the PLC's memory? Do these contact instructions operated differently, or identically?
- What happens when two coil instructions are linked to the same bit address in the PLC's memory, but driven to different states (e.g. one "energized" and the other "de-energized")?
- Does your PLC offer a special type of contact or other bit-level instruction to detect the *transistion* of a bit from one state to another? If so, how is this instruction used?
- Where in the PLC's memory are the single-bit registers (e.g. input registers, output registers, and internal bit registers) located? What symbol(s) are used to address each one?
- Where in the PLC programming editor can you view the "live" status of contact and coil bits?
- Experiment with using the *force* utility in your PLC to force certain bits to fixed values regardless of program operation. How will the operation of your program be affected if a particular input bit is forced? How will the operation of your program be affected if a particular output bit is forced? How can you tell from the live program display that bits have been forced to fixed values?

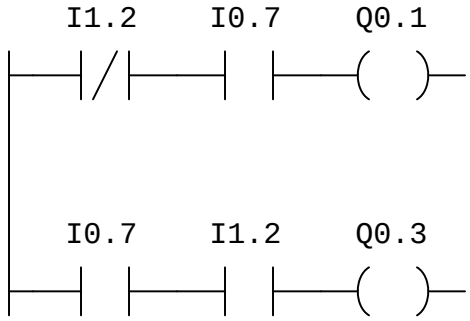
file i03354

Question 28

Suppose we have a Siemens S7-200 PLC connected to a pair of momentary-contact pushbutton switches and light bulbs as shown in this illustration:



Examine the following relay ladder logic (RLL) program for this Siemens PLC, determining the statuses of the two lamps provided both switches are simultaneously pressed by a human operator:

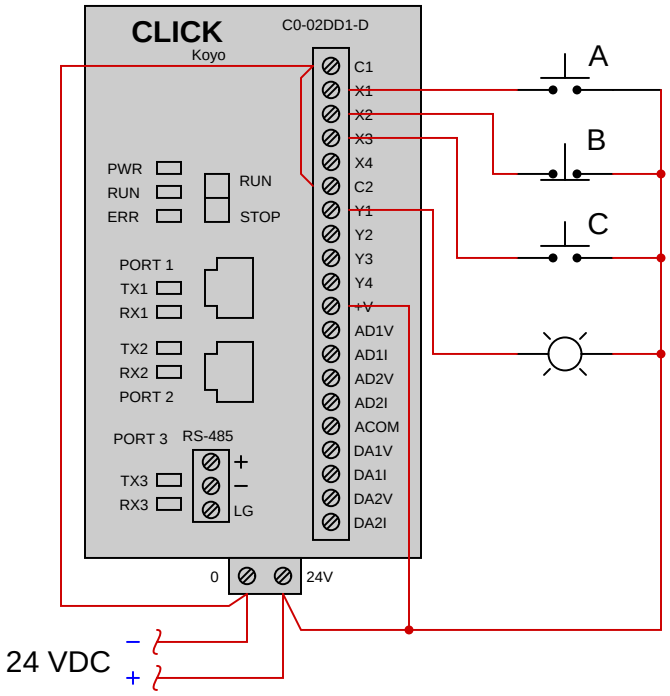


Finally, draw color highlighting showing how these "contact" instructions will appear in an online editor program given the stated input conditions.

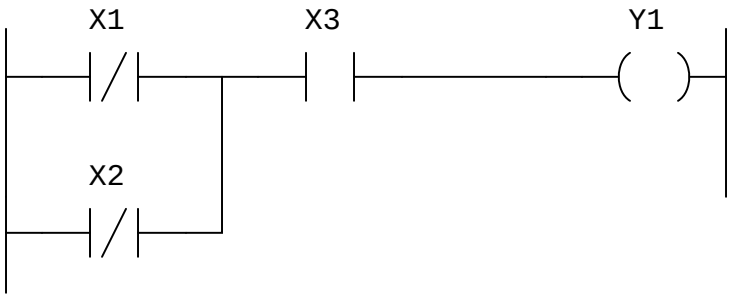
file i04664

Question 29

Suppose we have a Koyo "CLICK" PLC connected to three momentary-contact pushbutton switches as shown in this illustration:



Determine the necessary switch actuation statuses (i.e. pressed versus unpressed) to turn the lamp on given the following program running in the PLC:



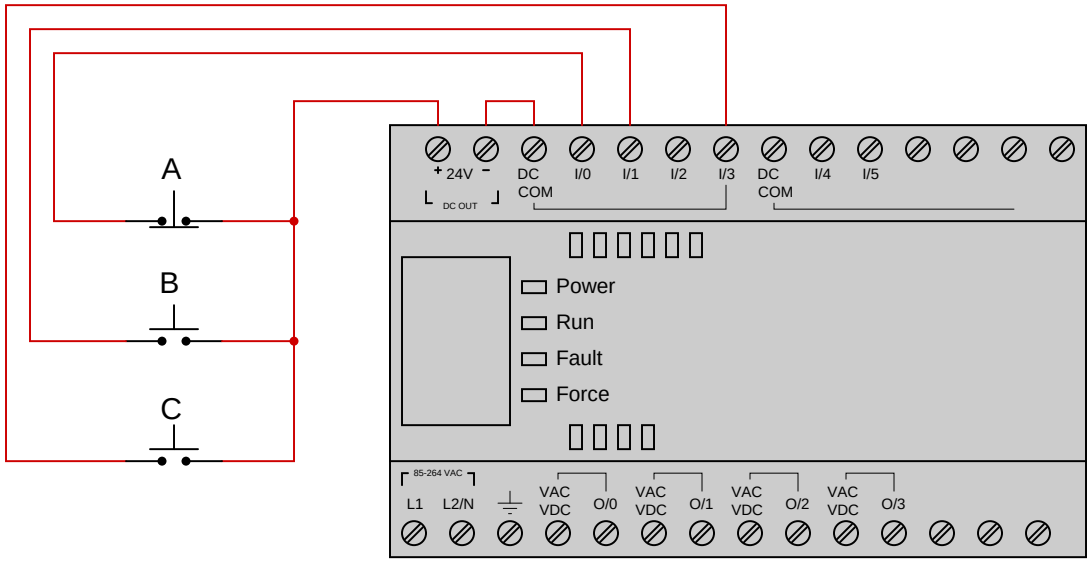
Suggestions for Socratic discussion

- Identify the significance of the labels "X" and "Y" for this PLC's bits. What do you suppose "X" signifies? What do you suppose "Y" signifies?

  file i04638

Question 30

Suppose we have an Allen-Bradley MicroLogix 1000 PLC connected to three momentary-contact pushbutton switches as shown in this illustration:

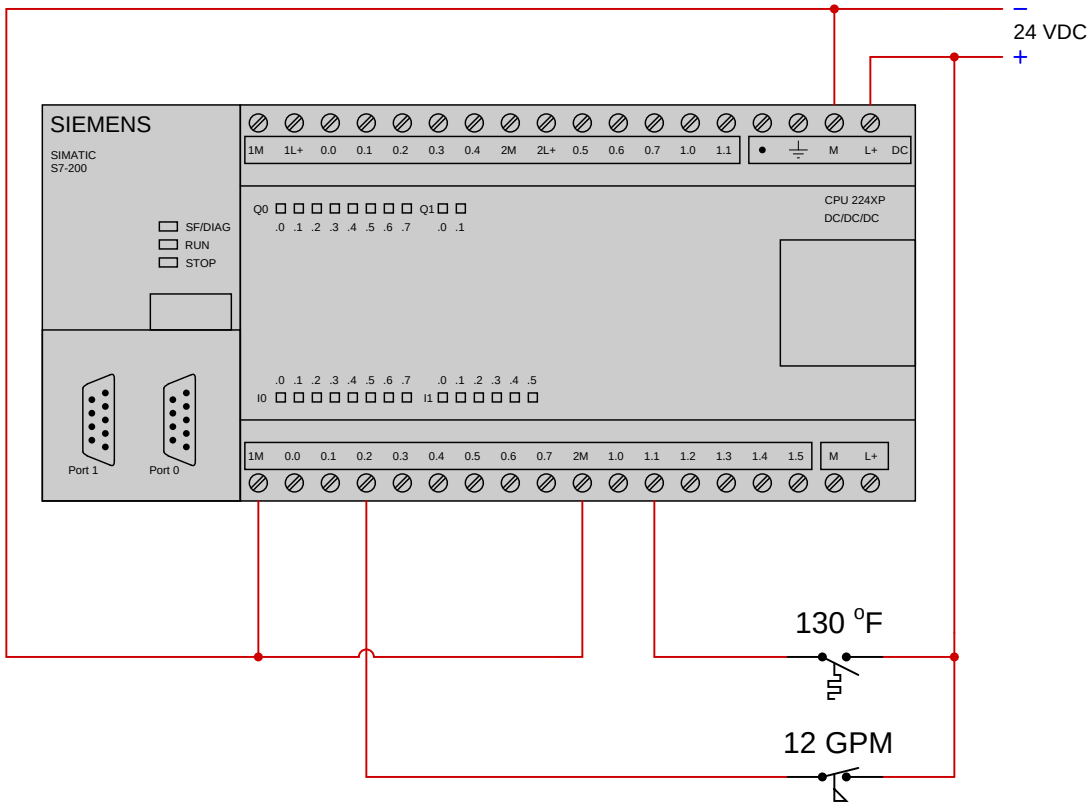


Determine the bit statuses of `I:0/0`, `I:0/1`, and `I:0/3` when switch A is pressed, switch B is unpressed (released), and switch C is pressed.

file i04685

Question 31

Suppose we have a Siemens S7-200 PLC connected to two process switches as shown in this illustration:

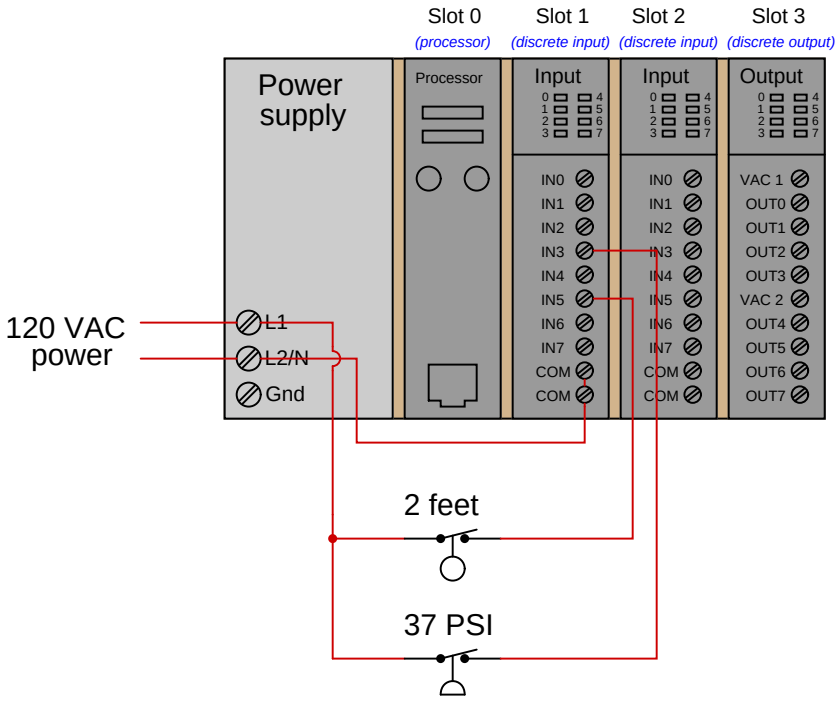


Determine the bit statuses of `I0.2` and `I1.1` when the temperature switch senses 122 °F and the flow switches senses 15 GPM.

file i04686

Question 32

Suppose we have an Allen-Bradley SLC 500 PLC connected to two process switches as shown in this illustration:

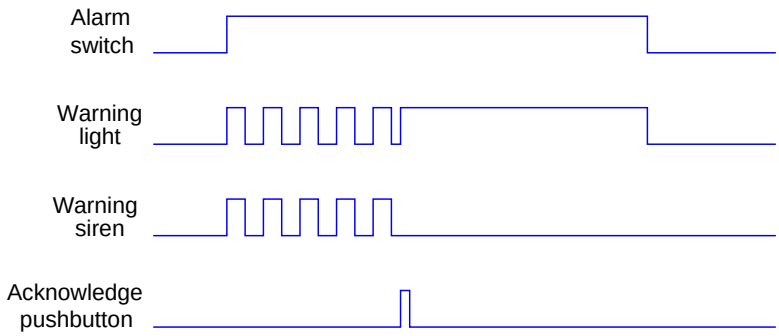


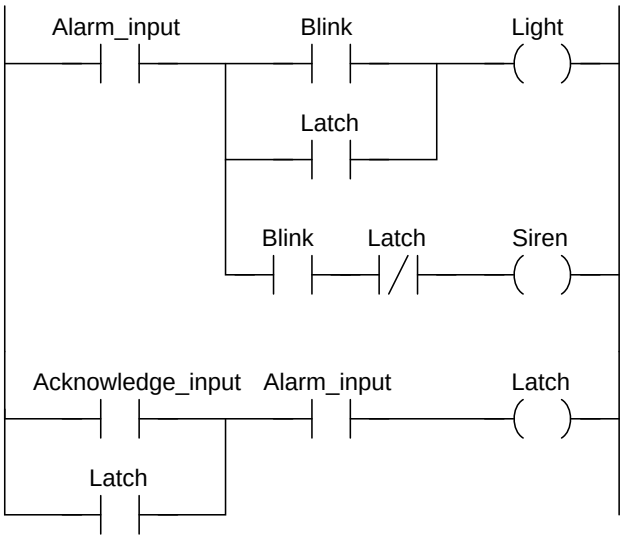
Determine the bit statuses of `I:1/3` and `I:1/5` when the level switch senses 3 feet and the pressure switch senses 14 PSI.

file i04687

Question 33

The following PLC program preforms the function of an *alarm annunciator*, where a discrete input signal from an alarm switch (e.g. high temperature alarm) first causes a warning light to blink and a siren to audibly pulse until a human operator presses an *acknowledge* pushbutton. If the alarm switch signal is still activated, the light will remain on (steady) instead of blink and the siren will go silent. The light turns off as soon as the alarm signal goes back to its "safe" state. A timing diagram shows how this should work:





Take this "generic" PLC program and enter it into your own PLC, assigning appropriate addresses to all instructions, and demonstrating its operation.

Suggestions for Socratic discussion

- Does the PLC program (as written) "expect" a *closed* alarm switch contact to trigger the alarm, or an *open* alarm switch contact?
- If the real-world alarm switch contact was a pressure switch wired NC (normally-closed), would this circuit function as a *low* pressure alarm or as a *high* pressure alarm?
- If the real-world alarm switch contact was a temperature switch wired NO (normally-open), would this circuit function as a *low* temperature alarm or as a *high* temperature alarm?

file i02342

Question 34

Programming Challenge and Comparison – Conveyor start/stop control with safety switch

Suppose we wish to control the starting and stopping of a large conveyor belt at a factory using a PLC. This control system will use a "Start" pushbutton, a "Stop" pushbutton, and an emergency shut-down pull-cable allowing anyone along the conveyor's length to stop the belt simply by tugging on a steel cable (this is akin to the "stop" cable used on public buses for passengers to signal to the driver their intent to get off at the next stop).

Inputs

- Start pushbutton (momentary NO) – *pushing this button closes the switch to energize the PLC input*
- Stop pushbutton (momentary NC) – *pushing this button opens the switch to de-energize the PLC input*
- Emergency stop cable (latching NC) – *tugging on the cable opens the switch to de-energize the PLC input*

Outputs

- Motor contactor – *energizing this PLC output starts the conveyor belt motor*

Write a PLC program performing this function, and demonstrate its operation using switches connected to its inputs to simulate the discrete inputs in a real application.

When your program is complete and tested, capture a screen-shot of it as it appears on your computer, and prepare to present your program solution to the class in a review session for everyone to see and critique. The purpose of this review session is to see multiple solutions to one problem, explore different programming techniques, and gain experience interpreting PLC programs others have written. When presenting your program (either individually or as a team), prepare to discuss the following points:

- Identify the "tag names" or "nicknames" used within your program to label I/O and other bits in memory
- Follow the sequence of operation in your program, simulating the system in action
- Identify any special or otherwise non-standard instructions used in your program, and explain why you decided to take that approach
- Show the comments placed in your program, to help explain how and why it works
- How you designed the program (i.e. what steps you took to go from a concept to a working program)

Suggestions for Socratic discussion

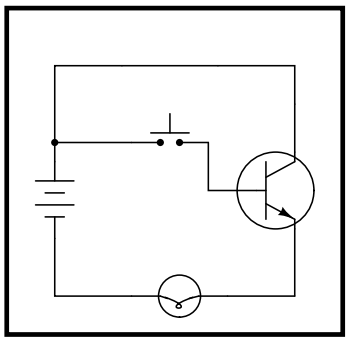
- How do you keep the motor "latched" on when the momentary "Start" switch is released?
- Which is simpler: implementing this function using normal program coils, or implementing this function using retentive coils ("set" and "reset", or "latch" and "unlatch")?

file i02340

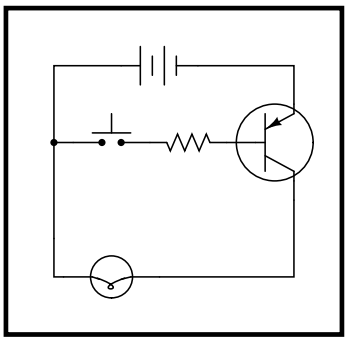
Question 35

Some of the following transistor switch circuits are properly configured, and some are not. Identify which of these circuits will function properly (i.e. turn on the load when the switch closes) and which of these circuits are mis-wired:

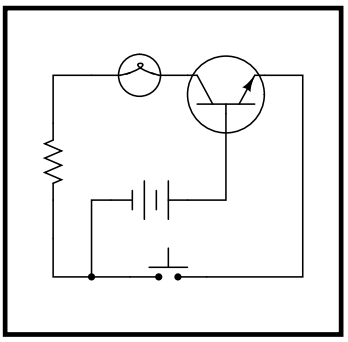
Circuit 1



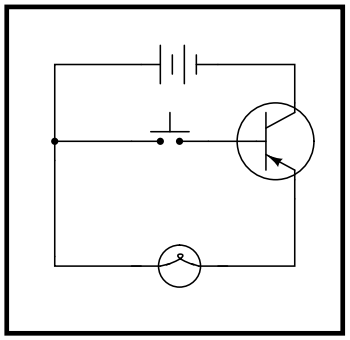
Circuit 2



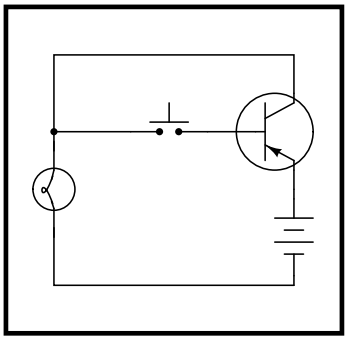
Circuit 3



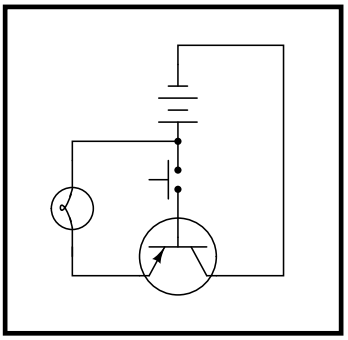
Circuit 4



Circuit 5



Circuit 6

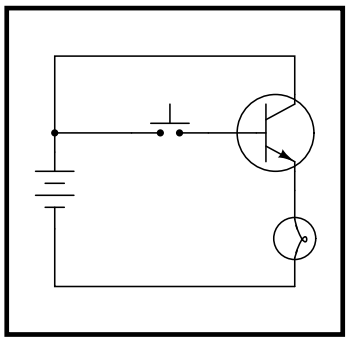


file i01004

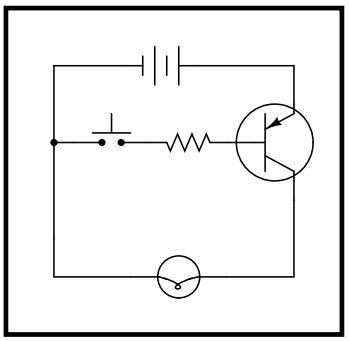
Question 36

In each of the following circuits, the light bulb will energize when the pushbutton switch is actuated. Assume that the supply voltage in each case is somewhere between 5 and 30 volts DC (with lamps and resistors appropriately sized):

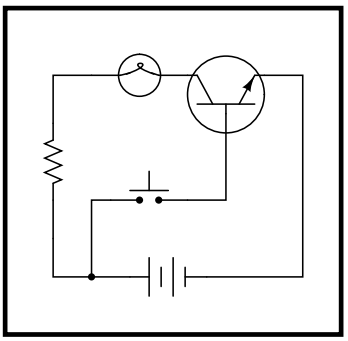
Circuit 1



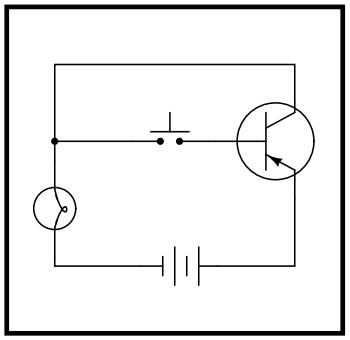
Circuit 2



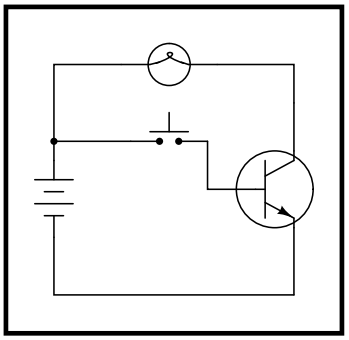
Circuit 3



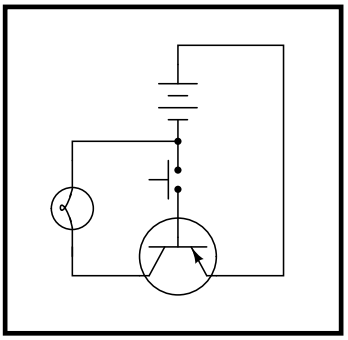
Circuit 4



Circuit 5



Circuit 6

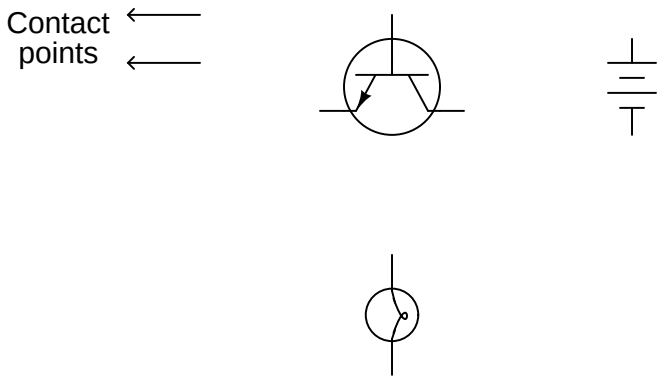


However, not all of these circuits are properly designed. Some of them will function perfectly, but others will function only once or twice before their transistors fail. Identify the faulty circuits, and explain why they are flawed.

file i01005

Question 37

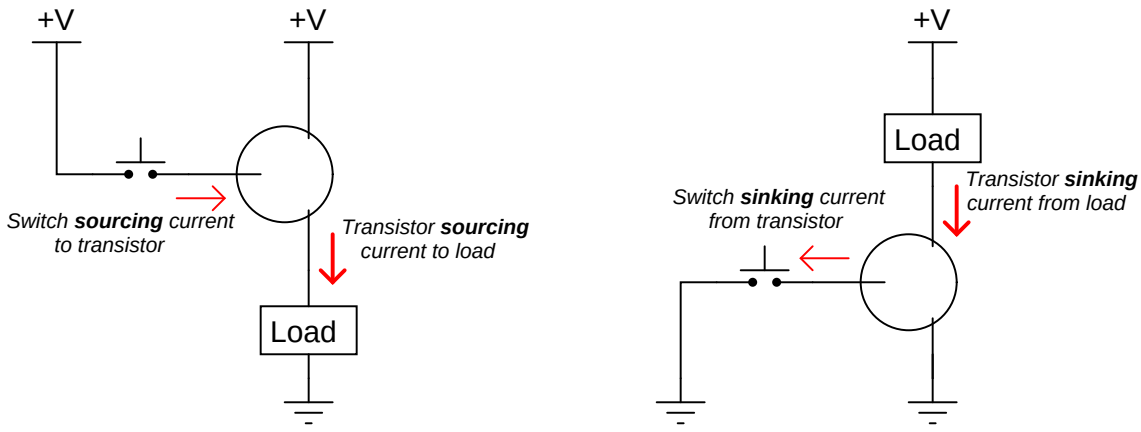
Draw the necessary wire connections so that bridging the two contact points with your finger (creating a high-resistance connection between those points) will turn the light bulb on:



file i01006

Question 38

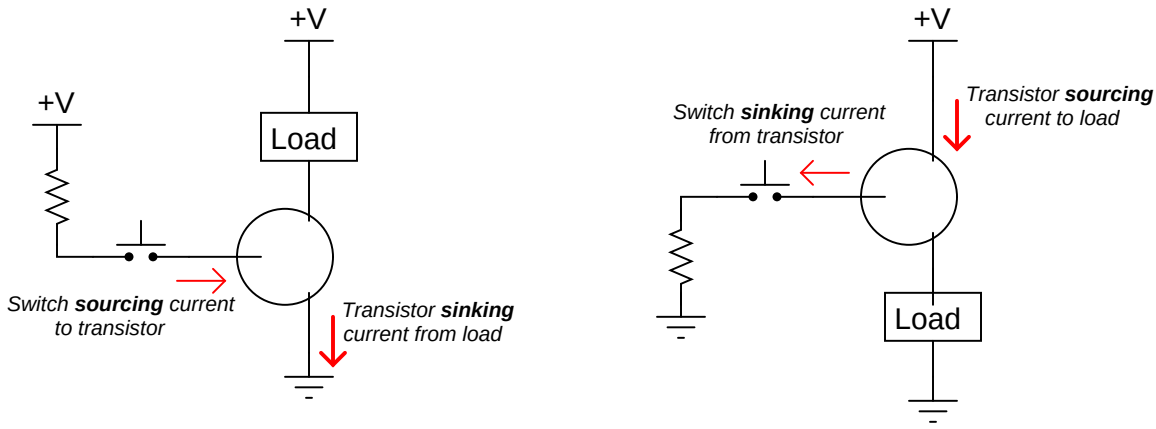
Choose the right type of bipolar junction transistor for each of these switching applications, drawing the correct transistor symbol inside each circle:



file i01007

Question 39

Choose the right type of bipolar junction transistor for each of these switching applications, drawing the correct transistor symbol inside each circle:

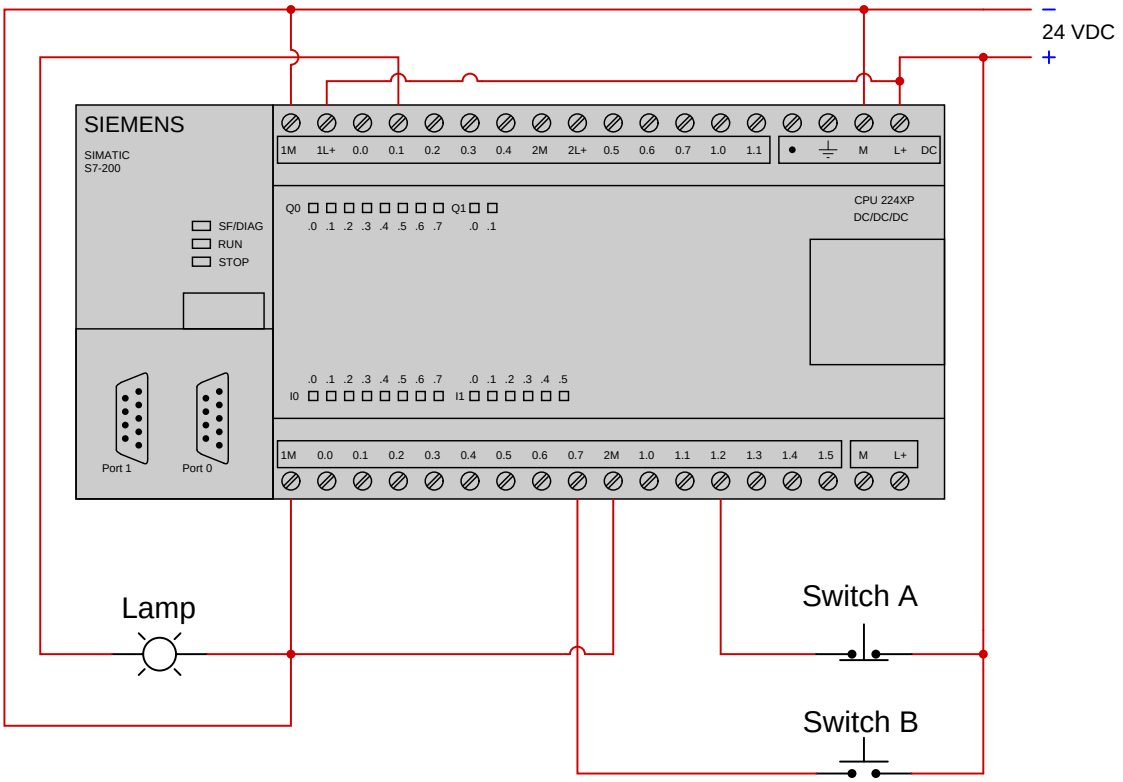


Also, explain why resistors are necessary in both these circuits for the transistors to function without being damaged.

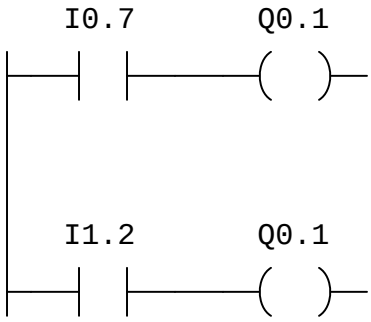
file i01008

Question 40

Suppose we have a Siemens S7-200 PLC connected to a pair of momentary-contact pushbutton switches and a light bulb as shown in this illustration:



Examine the following relay ladder logic (RLL) program for this Siemens PLC, determining the statuses of the two lamps provided both switches are simultaneously pressed by a human operator:



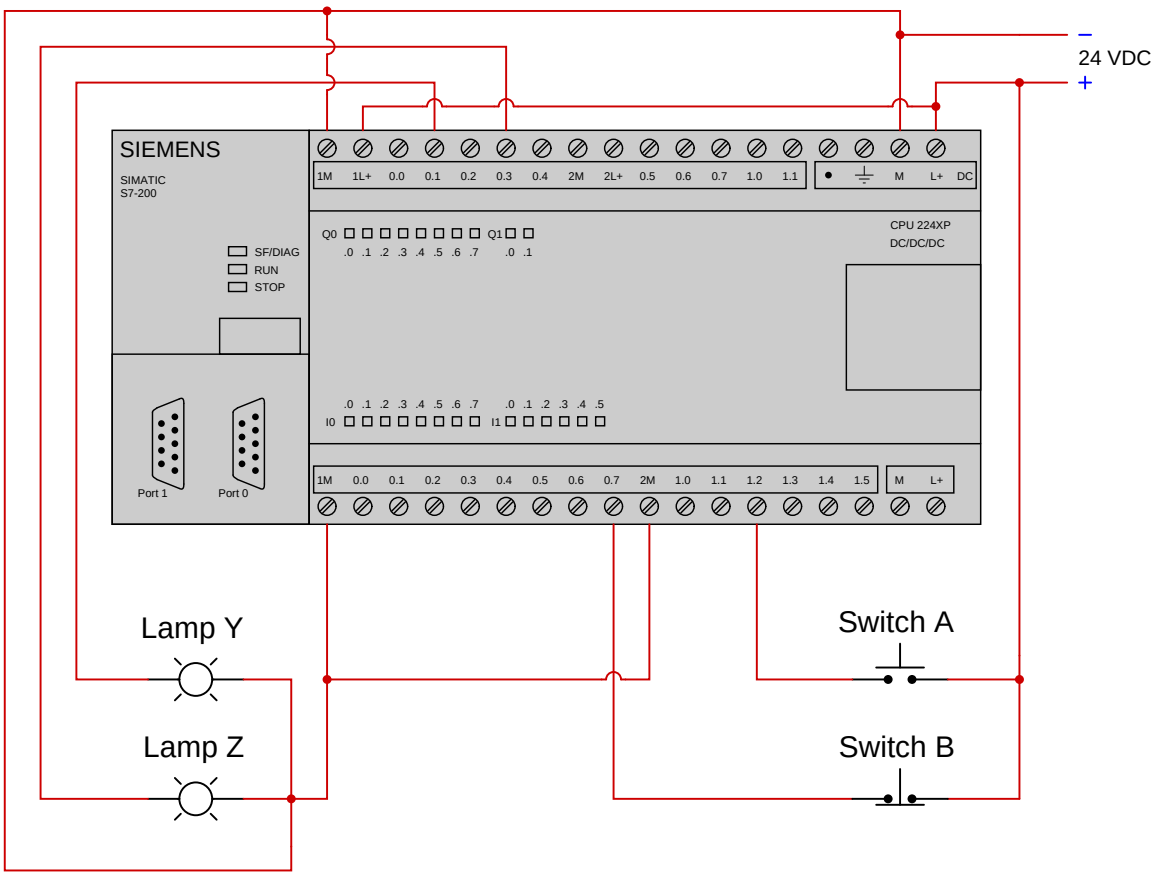
Complete the following "truth table" showing the status of the light bulb given all possible switch status combinations:

Switch A	Switch B	Light Bulb
Unpressed	Unpressed	
Unpressed	Pressed	
Pressed	Unpressed	
Pressed	Pressed	

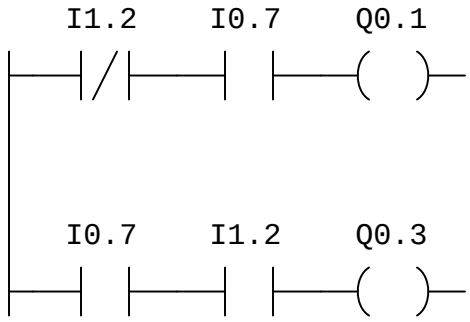
file i03360

Question 41

Suppose we have a Siemens S7-200 PLC connected to a pair of momentary-contact pushbutton switches and light bulbs as shown in this illustration:



Examine the following relay ladder logic (RLL) program for this Siemens PLC, determining the statuses of the two lamps provided switch A is pressed by a human operator and switch B is unpressed:

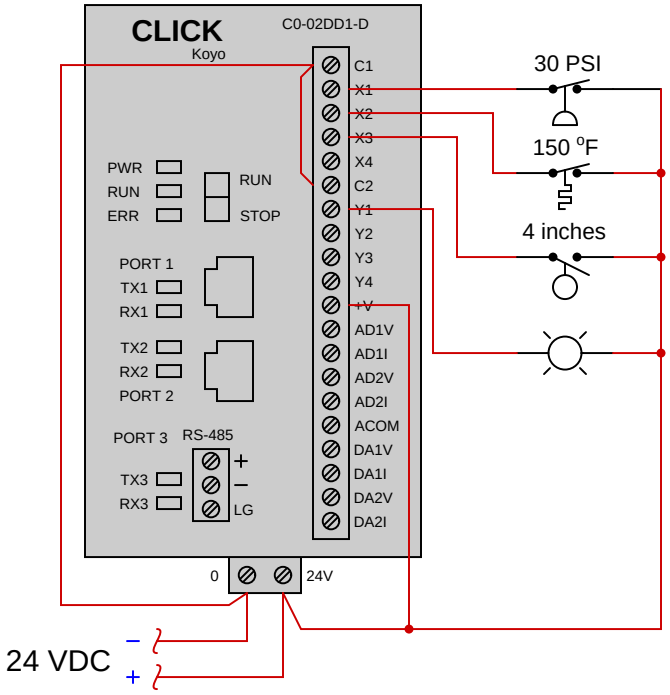


Furthermore, determine whether the inputs and outputs of this particular PLC (as shown) are *sourcing* or *sinking*.

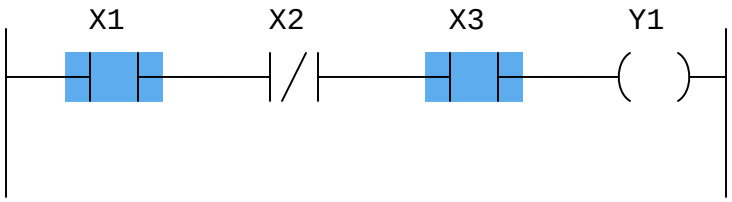
file i04170

Question 42

Suppose we have a Koyo "CLICK" PLC connected to three process switches as shown in this illustration:



Determine the switch stimuli (i.e. required pressure, temperature, and level) given the "live" display of the ladder logic program shown here:



Also, determine the status of the lamp connected to the PLC's `Y1` output.

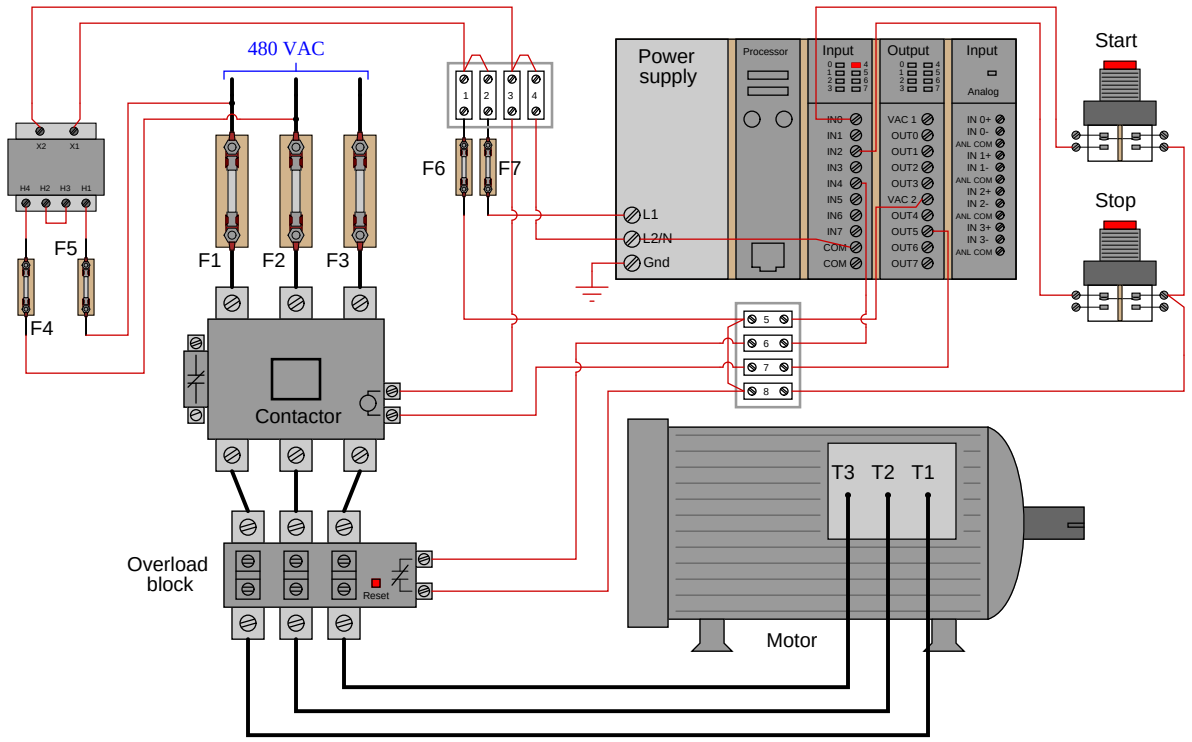
Suggestions for Socratic discussion

- Identify how you could override the PLC program to force the lamp to energize, if your only tool at hand was a screwdriver.

file i04667

Question 43

Suppose we have an Allen-Bradley SLC 500 controller connected to a pair of momentary-contact pushbutton switches and contactor controlling power to an electric motor as shown in this illustration:



This motor control system has a problem, though: the motor refuses to start when the "Start" pushbutton is pressed. Closely examine the pictorial diagram (including the status LEDs on the PLC's I/O cards), then identify at least two faults that could account for the motor's refusal to start.

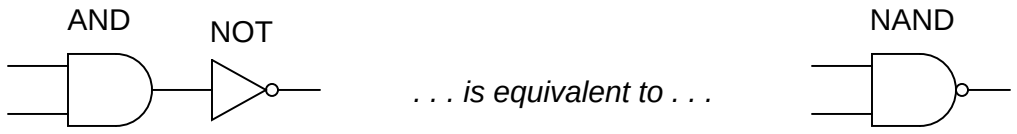
Suggestions for Socratic discussion

- A helpful problem-solving tip is to note the PLC's I/O states by examining the LED indicators on each input and output card on the PLC rack. What do the LED states tell you in this particular example?

  file i04069

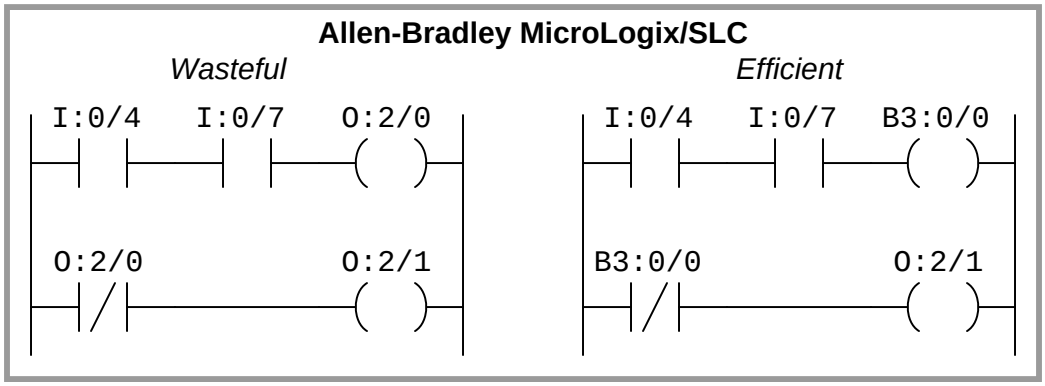
Question 44

A NAND logic function may be built up from a regular AND function plus an inverter function (a NOT gate) on the output:

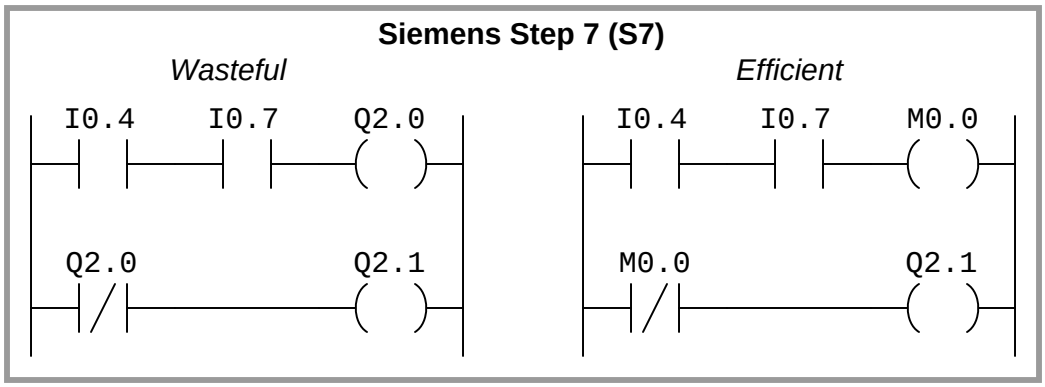


The same strategy of "building" a NAND gate may be done in PLC ladder-diagram programming, by combining a normally-closed contact instruction with two contacts in series.

Examine these two Allen-Bradley PLC programs, and explain why the left-hand program is "wasteful" while the right-hand program makes more efficient use of available bits:



Examine these two Siemens S7 PLC programs, and explain why the left-hand program is "wasteful" while the right-hand program makes more efficient use of available bits in the same ways the Allen-Bradley example programs were wasteful/efficient:



Note: many novice PLC programmers commit this error of "wasting" valuable I/O as they write their programs!

file i04092

Question 45

Read selected portions of the Siemens "SIMATIC S7-200 Programmable Controller System Manual" (document A5E00307987-04, August 2008) and answer the following questions:

Identify the different types of SIMATIC *counter* instructions.

Identify a practical application for a counter instruction programmed into a PLC.

How high can one of these counter instructions count up to? How low can it count down to? Based on these values, how many bits do you think are used in the register to store a counter instruction's current value?

Sketch a simple ladder-diagram program for a Siemens S7-200 PLC whereby a switch connected to input `I0.5` causes a counter to increment (count *up*) and then turn on an alarm light output `Q0.3` when the count reaches a value of 5. Also provide a "reset" function triggered by a normally-open switch contact at input `I0.0` to force the count value back to zero when pressed.

Suggestions for Socratic discussion

- If you have access to your own PLC for experimentation, I urge you to write a simple *demonstration* program in your PLC allowing you to explore the behavior of these PLC instructions. The program doesn't have to do anything useful, but merely demonstrate what each instruction does. First, read the appropriate section in your PLC's manual or instruction reference to identify the proper syntax for that instruction (e.g. which types of data it uses, what address ranges are appropriate), then write the simplest program you can think of to demonstrate that function in isolation. Download this program to your PLC, then run it and observe how it functions "live" by noting the color highlighting in your editing program's display and/or the numerical values manipulated by each instruction. After "playing" with your demonstration program and observing its behavior, write comments for each rung of your program explaining in your own words what each instruction does.

  file i02245

Question 46

Read selected portions of the Allen-Bradley "Logix5000 Controllers General Instructions" reference manual (publication 1756-RM0031-EN-P, January 2007) and answer the following questions:

Identify the different types of *counter* instructions offered in the Logix5000 PLC family.

How high can one of these counter instructions count up to? How low can it count down to? Based on these values, how many bits do you think are used in the register to store a counter instruction's current value?

Unlike the Siemens S7 family of PLCs, the Allen-Bradley counter instruction "box" symbols do not provide a place to connect a *reset* input. How then is it possible to command a counter instruction to reset back to zero?

Sketch a simple ladder-diagram program for an Allen-Bradley Logix5000 PLC whereby a high-temperature switch input with the tag-name `High_Motor_Temp` causes a counter to increment (count *up*) every time a motor overheats, and then turn on an alarm light output (tag-name `Alarm_Lamp`) when the count reaches a value of 5. Also provide a "reset" function triggered by a normally-open switch contact (tag-name `Alarm_Reset`) to force the count value back to zero when pressed.

Suggestions for Socratic discussion

- If you have access to your own PLC for experimentation, I urge you to write a simple *demonstration* program in your PLC allowing you to explore the behavior of these PLC instructions. The program doesn't have to do anything useful, but merely demonstrate what each instruction does. First, read the appropriate section in your PLC's manual or instruction reference to identify the proper syntax for that instruction (e.g. which types of data it uses, what address ranges are appropriate), then write the simplest program you can think of to demonstrate that function in isolation. Download this program to your PLC, then run it and observe how it functions "live" by noting the color highlighting in your editing program's display and/or the numerical values manipulated by each instruction. After "playing" with your demonstration program and observing its behavior, write comments for each rung of your program explaining in your own words what each instruction does.

  file i02664

Question 47

Demonstration Program – counter instructions

An important technique for learning any programming language – Ladder Diagram PLC programming included – is to write simple "demonstration" programs showcasing and explaining how particular instructions and programming constructs are supposed to work. Since you have access to your own personal PLC, you can explore the elements of your PLC's programming language like a scientist would explore new specimens: subject them to tests and record how they respond. This is how you will be able to teach yourself new models of PLC when you are working in your career, when you won't have textbooks to follow or training to show you exactly what to do.

Write such a "demonstration" program for your PLC's *counter* instructions, where discrete inputs on your PLC control discrete outputs on your PLC. An acceptable demonstration program must meet these three criteria:

- **Simple** – nothing "extra" included in the program to detract from the fundamental behavior of the instruction(s) being explored
- **Complete** – nothing missing from the program relevant to the fundamental behavior of the instruction(s) being explored. *For a counter demonstration program, this includes up counters, down counters, and up/down counters, all with provision for re-setting.*
- **Clearly documented** – every rung clearly commented in your own words, every variable named

Your instructor will challenge you to use this demonstration program to illustrate what you have learned about PLC counter instructions.

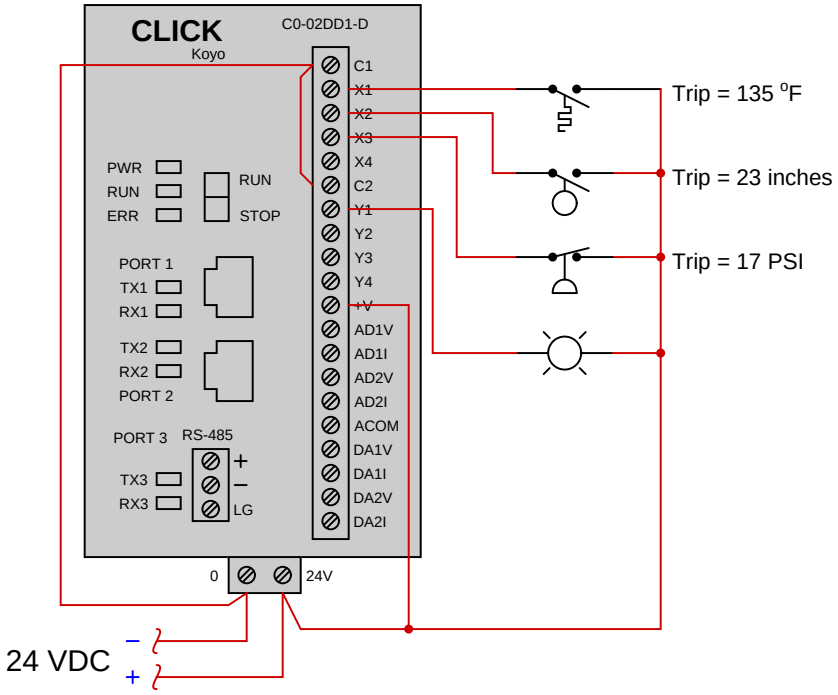
Suggested questions your demonstration program should answer:

- What are the different counter instruction types offered on your PLC? What does each one of them do?
- How can you make a single counter both *increment* (count up) and *decrement* (count down)?
- Where in the PLC programming editor can you view the "live" status of a counter instruction?
- Where in the PLC's memory are the counter variables (e.g. accumulated value, setpoint value) located? What symbol(s) are used to address each one?
- How far up can a counter count? How far down? Note that this will be related to the number of bits the counter instruction uses to track its current (accumulated) value.
- What happens when a counter reaches its preset value? How do you use this event to trigger something else to happen in the program?
- What happens to the counter's current value when it reaches its preset value? Does the counter stop counting, or does it continue counting past this threshold?
- When a counter is reset, does its current value begin at zero or one?
- Is it possible to "preload" a counter instruction so that it doesn't have to begin at the starting value when the PLC program runs anew?
- What happens to the counter's current value when it reaches its maximum value? Does the counter instruction stop counting, or does it do something else?

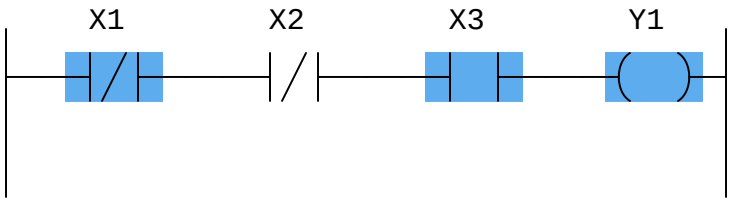
file i03353

Question 48

Suppose we have a Koyo "CLICK" PLC connected to three process switches as shown in this illustration:



Determine the process conditions (i.e. temperature, level, and pressure values) given the "live" display of the ladder logic program shown here:

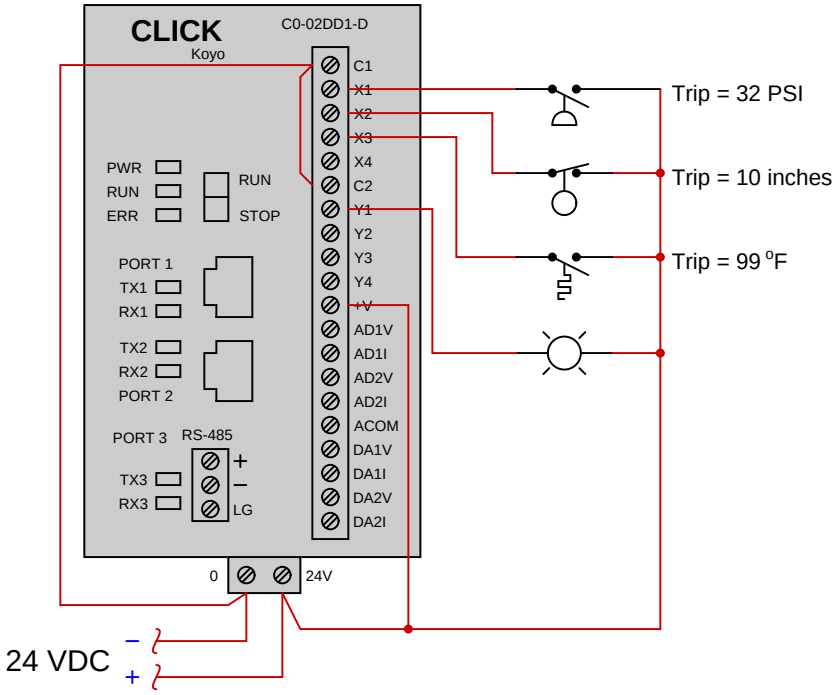


Also, determine the status of the lamp connected to the PLC's `Y1` output.

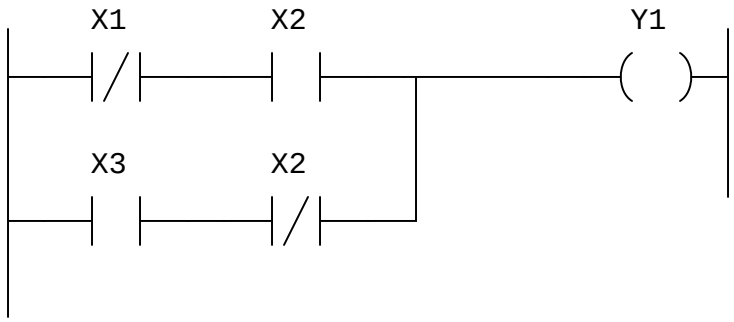
file i02144

Question 49

Suppose we have a Koyo "CLICK" PLC connected to three process switches as shown in this illustration:



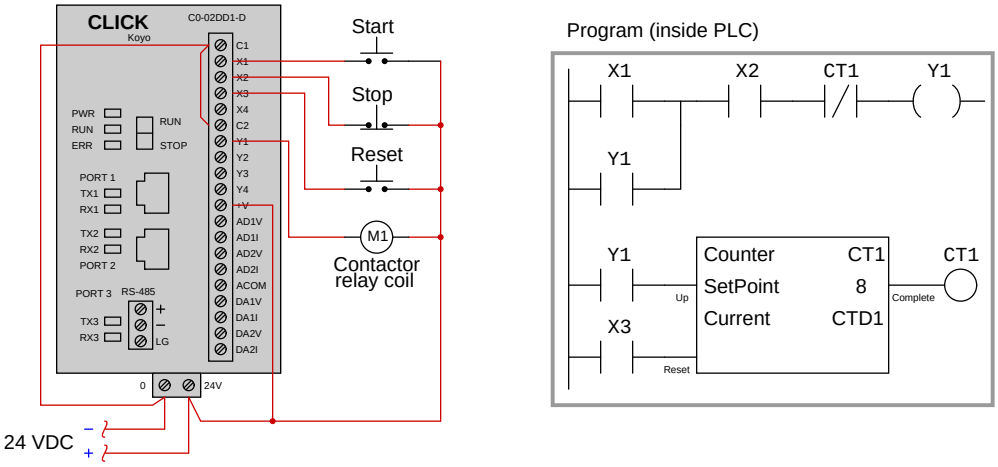
Determine the process conditions (i.e. temperature, level, and pressure values) given the "offline" display of the ladder logic program shown here, knowing that the lamp happens to be *energized* at the present time:



file i02145

Question 50

This Koyo "CLICK" PLC has been programmed to control the starting and stopping of an electric motor, including a *counter* instruction to prevent the motor from being started up more than a specified number of times:



Identify the counter instruction in the program shown, its input "connections", and also how the result of the counter reaching its pre-set limit forces the motor to stop. Also, determine the maximum number of times the motor may be started up, assuming the counter's current value goes to zero when the Reset button is pressed.

Finally, determine how to modify this PLC program so that the counter may be manually reset by the operator without requiring a separate pushbutton labeled "Reset".

Suggestions for Socratic discussion

- If an operator presses the "Start" button multiple times while the motor is already running, do these button-presses get counted by the counter instruction, or do only the real motor start-up events get counted?
- What do you suppose the label "CTD1" represents inside the counter instruction?
- Note the number of times the bit `Y1` is referenced inside this PLC program: once in a coil instruction and twice in contact instructions. Is there any limit to how many times a bit address may be used in a PLC program?
- Describe the purpose of the first contact instruction labeled `Y1` in this program, explaining why it is often referred to as a *seal-in* contact.

file i03589

Question 51

Programming Challenge – Parking garage counter

Suppose we wish to count the number of cars inside a parking garage at any given time, by incrementing a counter each time a car enters the garage through the entry lane, and decrementing the same counter each time a car leaves the garage through the exit lane. One discrete input of the PLC will connect to a switch detecting the passing of each car through the garage entry, and another discrete input of the PLC will connect to a switch detecting cars passing out the garage exit. The PLC must be equipped with a way to for the garage attendant to manually reset the counter to zero.

Write a PLC program to perform this function, and demonstrate its operation using switches connected to its inputs to simulate the discrete inputs in a real application.

Suggestions for Socratic discussion

- What type of switches would you recommend to detect cars driving into the parking garage?
- How are you able to view the counter instruction's current count value as the program runs?
- Is there any way to "fool" this system so that it does not hold an accurate count of cars inside the garage?

PLC comparison:

- Allen-Bradley Logix 5000: `CTUD` count-up/down instruction
- Allen-Bradley SLC 500: `CTU` and `CTD` instructions.
- Siemens S7-200: `CTUD` count-up/down instruction
- Koyo (Automation Direct) DirectLogic: `UDC` counter instruction

file i03684

Question 52

Question 53

Question 54

Question 55

Question 56

Question 57

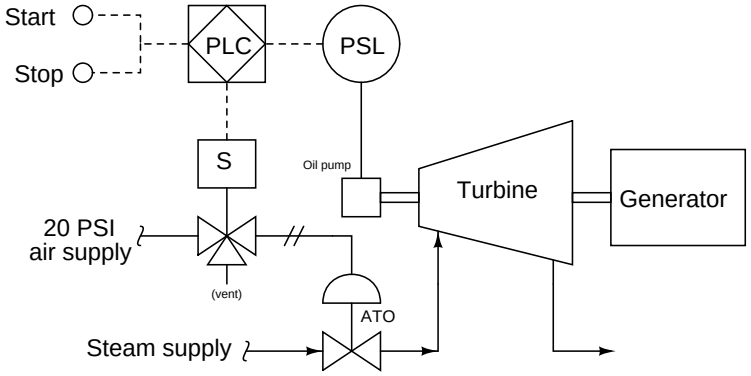
Question 58

Question 59

Question 60

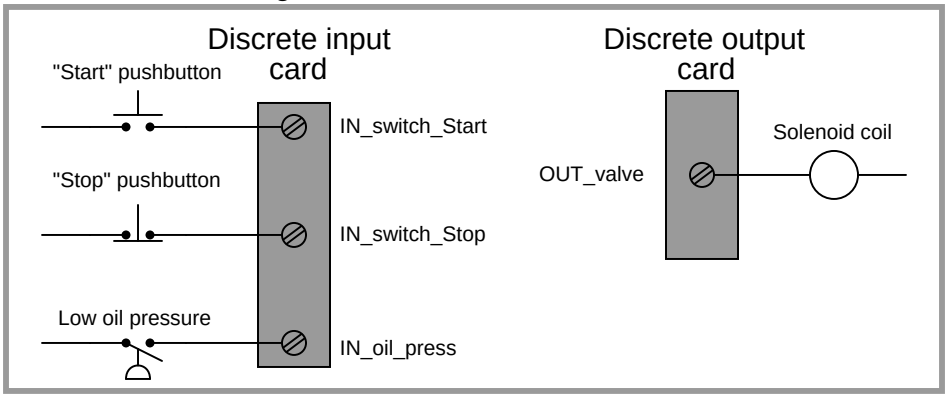
Question 61

A PLC is being used to monitor the oil pressure for a steam turbine driving an electrical generator, shutting steam off to the turbine if ever the oil pressure drops below a 10 PSI limit. The turbine's lubrication oil pump is driven by the turbine shaft itself, supplying itself with pressurized lubricating oil to keep all the turbine bearings properly lubricated and cooled:

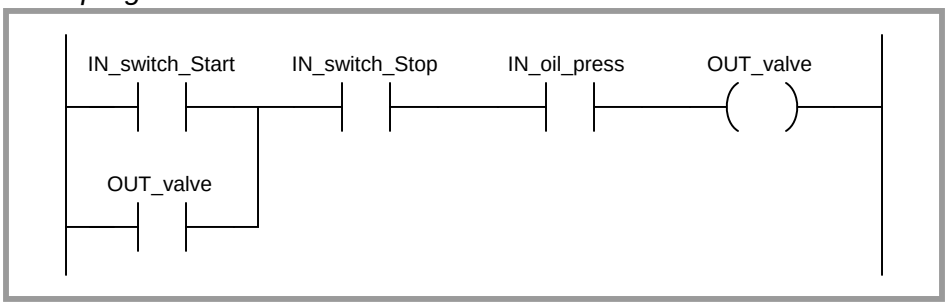


Another technician programmed the PLC for the start/stop function, but this program has a problem:

Real-world I/O wiring



PLC program

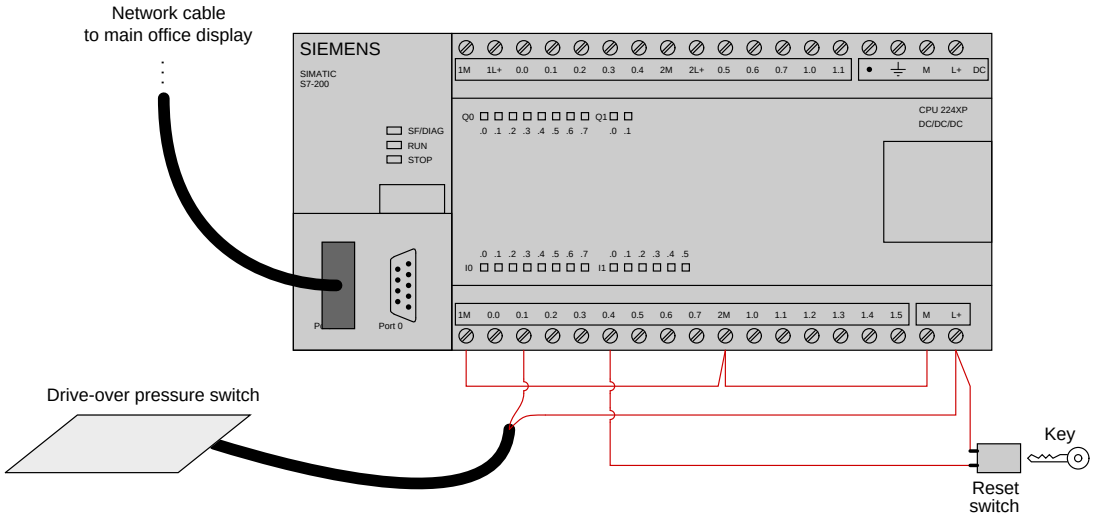


Identify what this problem is, and fix it! Hint: the oil pump is driven by the turbine, and as such cannot generate any oil pressure until the turbine begins to spin.

file i00189

Question 62

This Siemens S7-200 PLC is supposed to count the number of cars entering a parking garage, using a pressure-sensitive switch that the cars drive over when entering the garage. The car-count value is sent to a computer in the main office via a network cable plugged into the PLC. The parking attendant is able to reset the count to 0 at the end of his shift, using a key-switch:



Unfortunately, there is something wrong with this system. Although it worked just fine yesterday, today the counter's current value as displayed on the main office computer seems to be stuck at 574 no matter how many more cars drive over the pressure switch and enter the garage. Explain how you would go about diagnosing the problem in this system, justifying each step you would take.

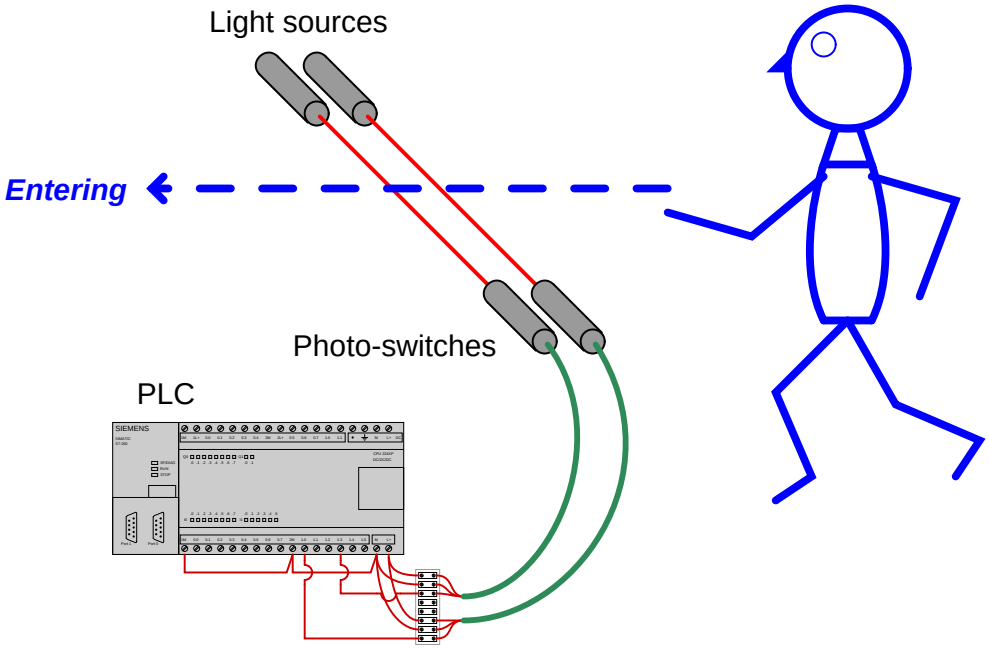
Suggestions for Socratic discussion

- A useful troubleshooting strategy is to mentally divide this system into three major portions, and try to determine which portion the problem lies within: (1) the switches and wiring connected to the PLC, (2) the PLC itself, and (3) the network cable and computer in the main office.
- How important is the fact that this system worked fine yesterday? Does this knowledge help you in your troubleshooting?
- Are there any LED indicators on the face of the PLC that might be helpful in providing diagnostic data for you to pinpoint the location of the problem?

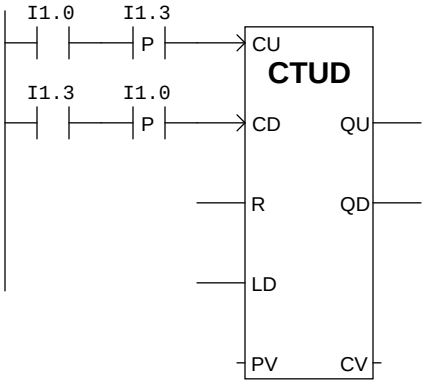
file i03683

Question 63

This Siemens S7-200 PLC has been programmed to count the number of people in a room, by incrementing a counter every time a person enters through the doorway, and decrementing that same counter whenever someone exits through the same doorway. The two optical switches activate whenever their respective light beams are broken by someone passing through. Their horizontal separation is just a couple of inches – much less than the girth of a person's torso. The operating status of each switch is that it energizes the PLC input when the light beam is broken:



Examine the program in this PLC for counting people, and determine how it is able to differentiate between a person entering the room and a person leaving the room:



Suggestions for Socratic discussion

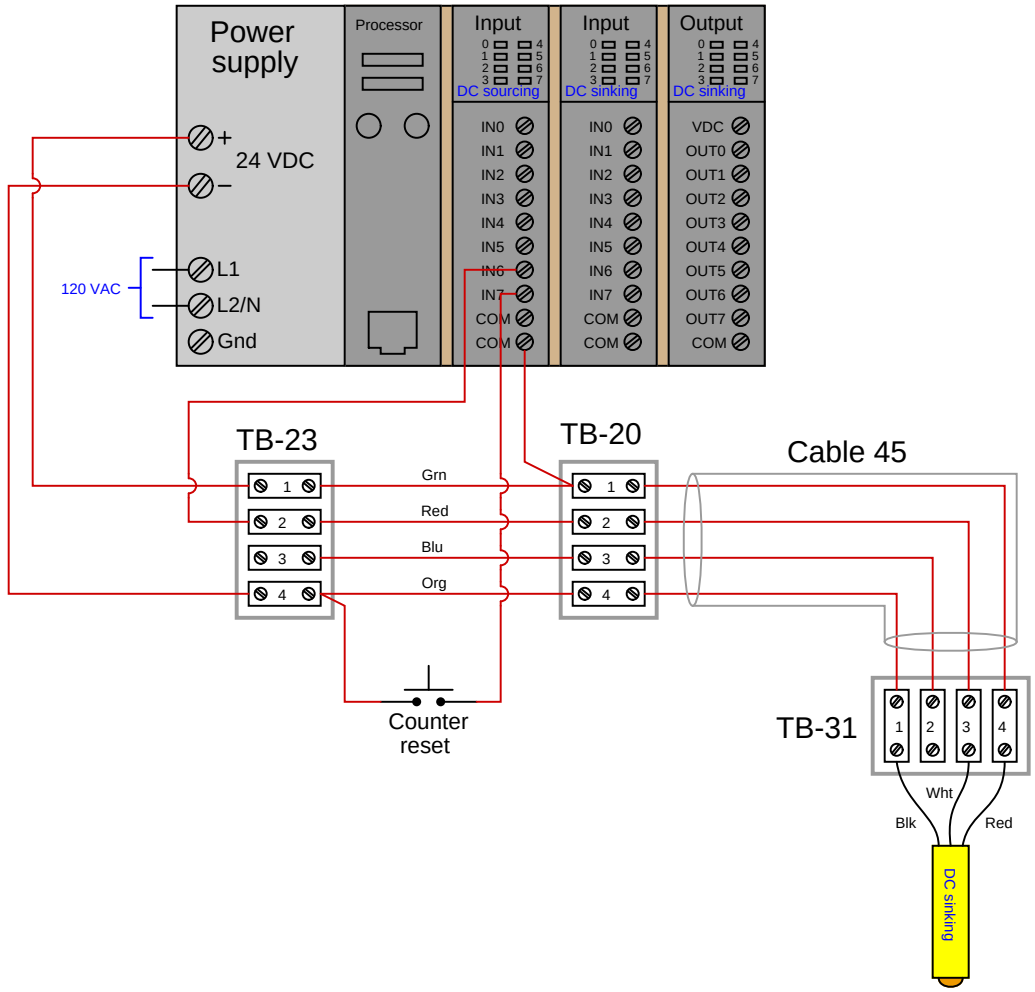
- Explain how a *timing diagram* of the switch states would be helpful in analyzing the operation of this PLC program.
- *Transition* (edge-detecting) functions are implemented in Allen-Bradley PLCs using the *one-shot rising* (OSR) instruction. Research how the OSR instruction is used, and how it differs from the "P" and "N" contacts shown in this Siemens PLC program.

- Will this system still function properly if the optical sensors are spaced farther apart than the width of a human body? Explain why or why not.

  file i00185

Question 64

A PLC is used to count the number of cans traveling by on a conveyor belt in a fish canning factory. An optical proximity switch detects the passage of each can, sending a discrete (on/off) signal to one of the PLC's input channels. The PLC then counts the number of pulses to determine the number of cans that have passed by:



One day the canning line operator tells you the PLC has stopped counting even though cans continue to run past the proximity switch as the conveyor belt moves. Identify what you would do to begin diagnosing this problem, justifying each step you would take.

Suggestions for Socratic discussion

- Identify different areas or components within this system that could possibly be at fault, as a prelude to identifying specific diagnostic steps.
- Are there any ways you could diagnose this problem without the use of test equipment (e.g. multimeter)?
- Explain the significance of the "sourcing" and "sinking" labels on the I/O cards as well as the proximity switch.

file i02428

Question 65

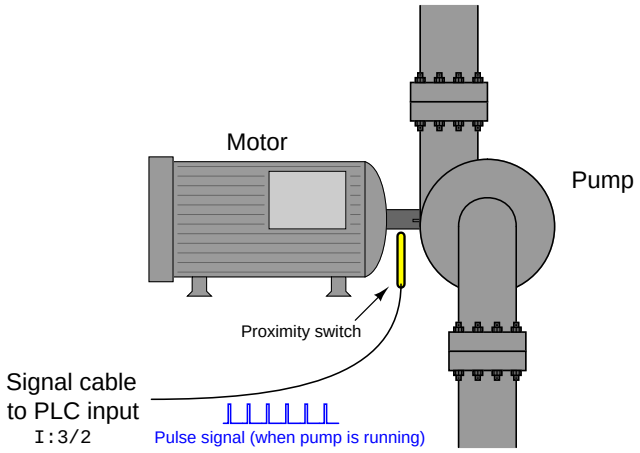
The manufacturing company you work for installs a PLC control system on its assembly line, counting the number of components produced every shift. For quite a while, the system works without any problems whatsoever, and then one day management decides to scrap a run of product mid-shift and start over. This is when they discover the system integrator they contracted to build and program the PLC system provided no way to reset the shift production counter except to wait until the shift is over.

An operations manager summons you to reset the counter for them. Identify at least two different ways you could reset the counter to meet their needs, as quickly as possible.

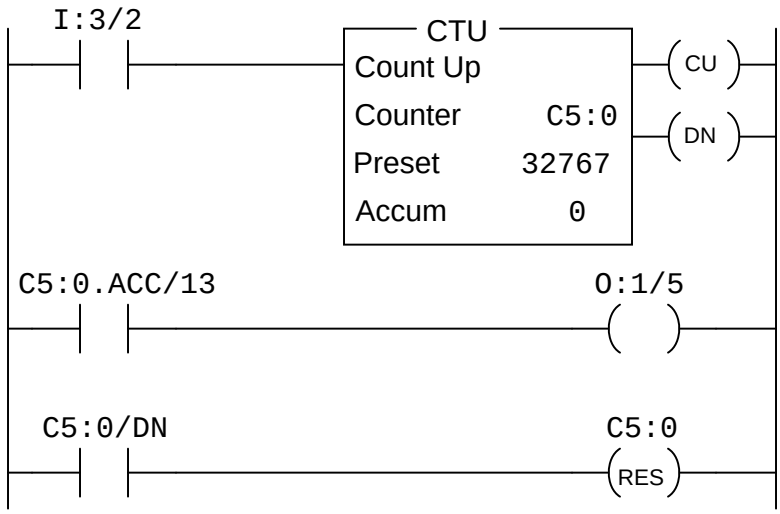
file i00182

Question 66

An important pump in a chemical process is turned by an electric motor, and operators want to have visual indication in the control room that the pump is indeed turning. There is no way to attach a speed switch to the pump shaft (that would be too easy!). Instead, someone has installed a proximity switch near the pump shaft, situated to pick up the passing of a keyway in the shaft with each rotation. Thus, the proximity switch will output a "pulse" signal when the pump shaft is spinning:



Operators wanted the indicator light in the control room to blink when the pump is running, for an indication of shaft motion. The problem is, the shaft turns much too fast (approximately 1750 RPM) to directly drive the indicator with the proximity switch signal, and so an Allen-Bradley PLC was programmed to produce a slower blink using this program:



Explain how this program works to fulfill the function of a *frequency divider*, converting the high-speed pulse signal of the proximity switch into a low-speed blink for the operator light.

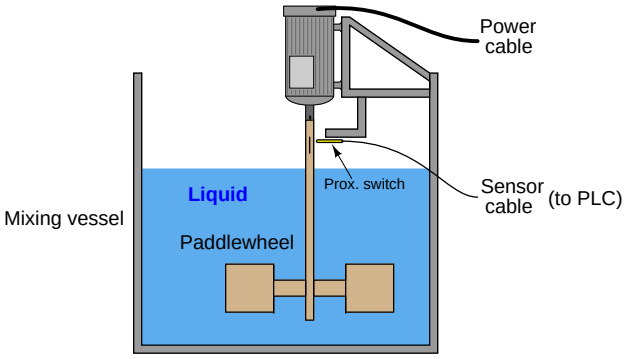
Suggestions for Socratic discussion

- Explain how a *frequency divider* circuit built out of J-K flip-flop integrated circuits functions, and then describe how this PLC program is similar in principle.
- Explain how to speed up the blinking rate of the light for any given motor shaft speed.

Question 67

Programming Challenge and Comparison – Mixer motor auto-stop

A batch mixing process in a manufacturing facility uses a mixer motor and a large "paddlewheel" to mix liquid ingredients to make a final product. A PLC needs to run this motor for exactly 1500 turns of the paddlewheel and then automatically stop. The motor needs to be able to start back up if the "Start" button is pressed again for the next batch:



Inputs

- Start pushbutton (momentary NO) – *pushing this button closes the switch to energize the PLC input*
- Stop pushbutton (momentary NC) – *pushing this button opens the switch to de-energize the PLC input*
- Proximity switch (NO) – *one pulse per paddle revolution*

Outputs

- Motor contactor – *energizing this PLC output starts the mixing motor*

Write a PLC program performing this function, and demonstrate its operation using switches connected to its inputs to simulate the discrete inputs in a real application.

When your program is complete and tested, capture a screen-shot of it as it appears on your computer, and prepare to present your program solution to the class in a review session for everyone to see and critique. The purpose of this review session is to see multiple solutions to one problem, explore different programming techniques, and gain experience interpreting PLC programs others have written. When presenting your program, prepare to discuss the following points:

- Identify the "tag names" or "nicknames" used within your program to label I/O and other bits in memory
- Follow the sequence of operation in your program, simulating the system in action
- Identify any special or otherwise non-standard instructions used in your program, and explain why you decided to take that approach
- Show the comments placed in your program, to help explain how and why it works
- How you designed the program (i.e. what steps you took to go from a concept to a working program)

Suggestions for Socratic discussion

- How can you test your program's basic operation without having to flick a switch *1500 times* to simulate the full number of paddle revolutions?
- Try writing your program so that the number of paddle-turns (1500) is not "hard-coded" into the PLC program, but rather resides in some memory location that may be altered without reprogramming the PLC.

file i03688

Question 68

Programming Challenge – Hour/Minute/Second timer

Many PLCs provide a range of *special contacts* to the programmer. Among these "special contacts" is typically one that cycles on and off at a rate of once per second, like a 1 Hz clock pulse.

Research the special contact for this function in your PLC, then write a PLC program for an Hour/Minute/Second timer using three counter instructions: one to count seconds (0 to 59), one to count minutes (0 to 59), and one to count hours.

Suggestions for Socratic discussion

- What is the address of the special contact in your PLC for the 1 Hz clock pulse?
- How do you make three counters count in the correct sequence, so that one represents seconds, the next minutes, and the next hours?

PLC comparison:

- Allen-Bradley SLC 500: status bit `S:4/0` is a free-running clock pulse with a period of 20 milliseconds, which may be used to clock a counter instruction up to 50 to make a 1-second pulse (because 50 times 20 ms = 1000 ms = 1 second).
- Siemens S7-200: Special Memory bit `SM0.5` is a free-running clock pulse with a period of 1 second.
- Koyo (Automation Direct) DirectLogic: Special relay `SP4` is a free-running clock pulse with a period of 1 second.

  file i03691

Question 69

Question 70

Question 71

Question 72

Question 73

Question 74

Question 75

Question 76

Question 77

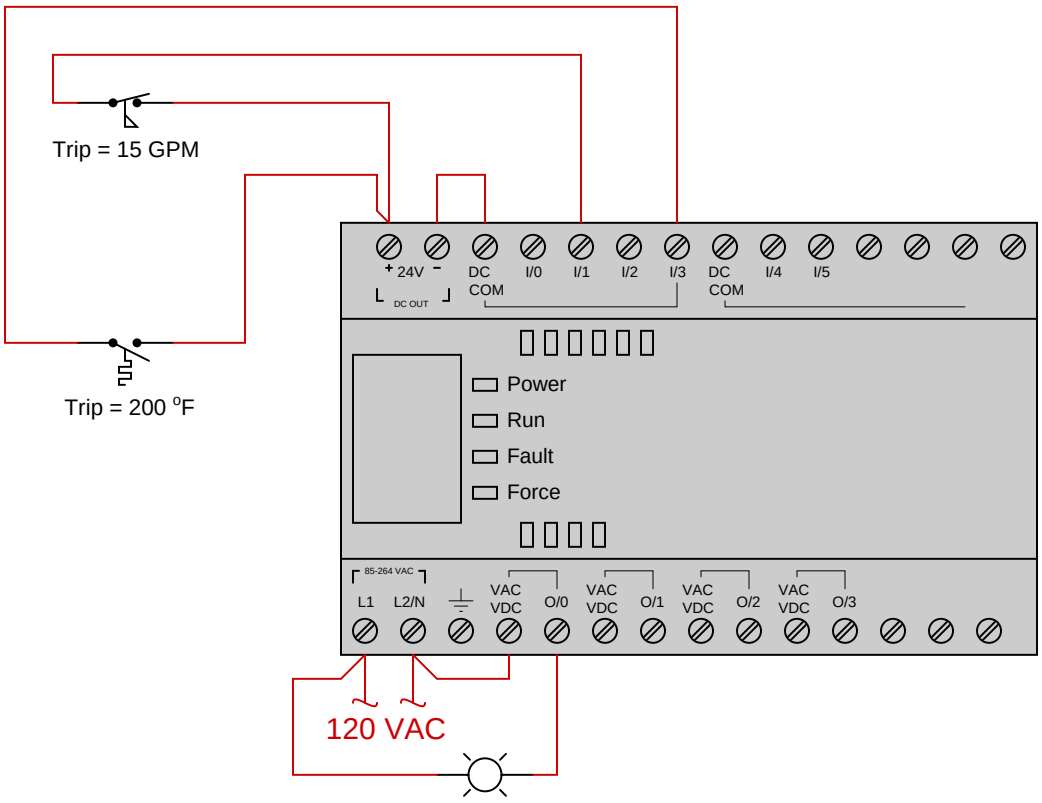
Question 78

Question 79

Question 80

Question 81

Suppose we have an Allen-Bradley MicroLogix 1000 PLC connected to a temperature switch and a flow switch:

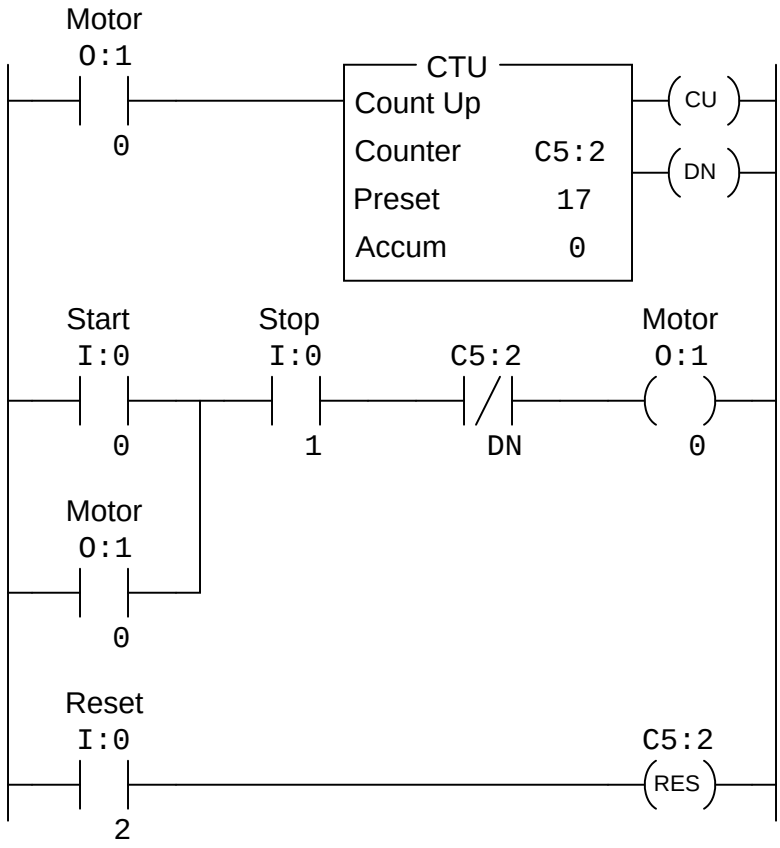


We wish for the lamp to come on when the temperature is below 200 degrees F *and* when the flow rate is below 15 GPM. Write a RLL program for the PLC (complete with correct address labels for each of the virtual contacts) to fulfill this function:

file i02375

Question 82

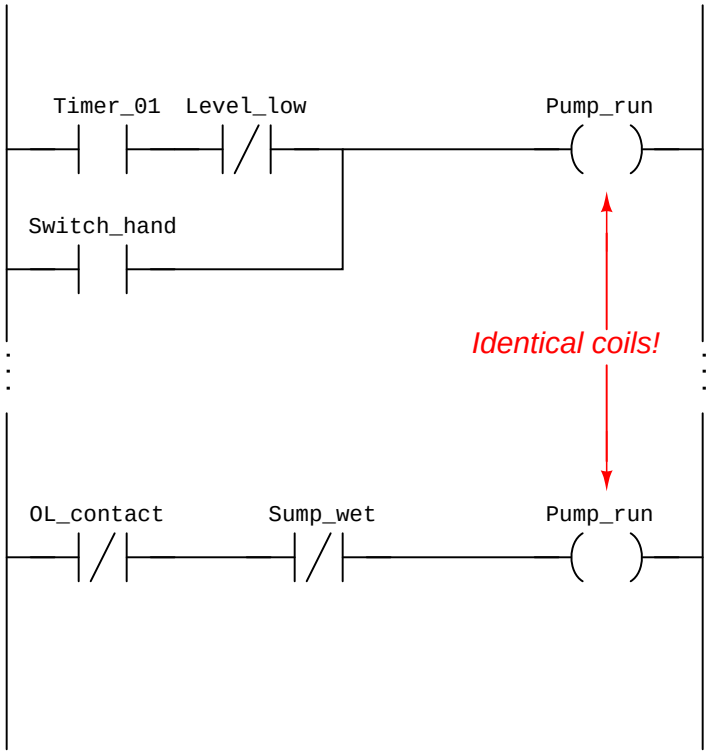
Analyze this Allen-Bradley PLC program and explain what it is supposed to do:



file i02377

Question 83

In relay ladder logic (RLL) programming, it is considered bad practice to have multiple instances of an identical (standard) "relay" coil in a program:



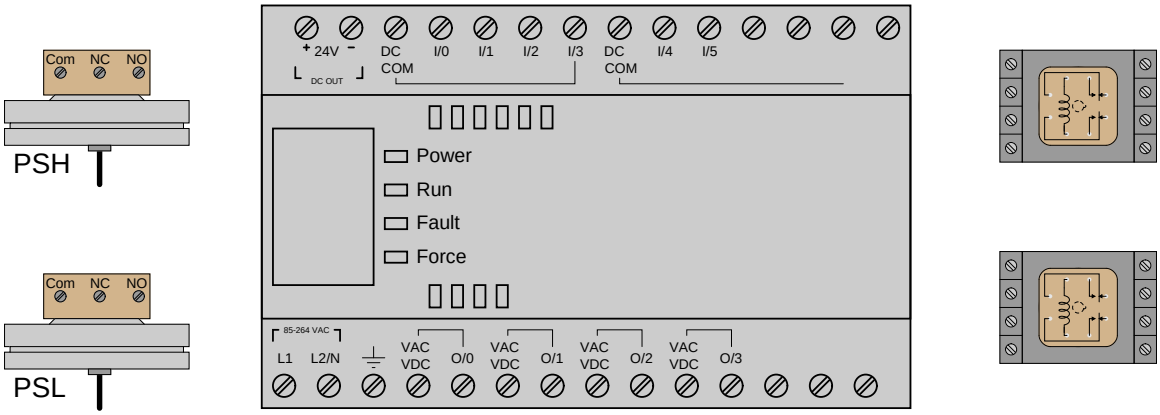
Explain why this is considered poor practice in PLC programming. Next, determine the status of the `Pump_run` output channel given the following bit states:

- `Timer_01` = 1
- `Level_low` = 1
- `Switch_hand` = 0
- `OL_contact` = 0
- `Sump_wet` = 0

file i02376

Question 84

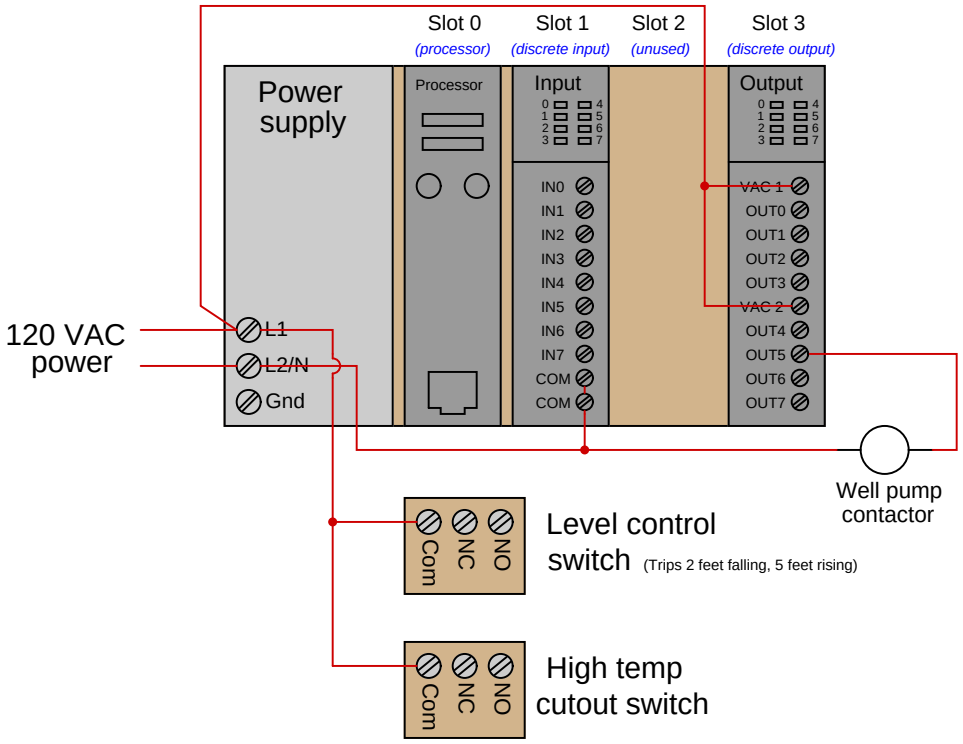
Sketch the wires necessary to connect two pressure switches and two relay coils to the following Allen-Bradley MicroLogix 1000 PLC (model 1761-L10BWA, with 6 discrete DC inputs either sourcing or sinking, and 4 discrete relay contact outputs). Be sure to wire the two switches so they *source* current to the PLC's inputs (the low-pressure switch to `I/2` and the high-pressure switch to `I/5`, normally-open contacts on both) and wire the relay coils so the PLC *sources* current to them (`O/0` and `O/1`):



file i02379

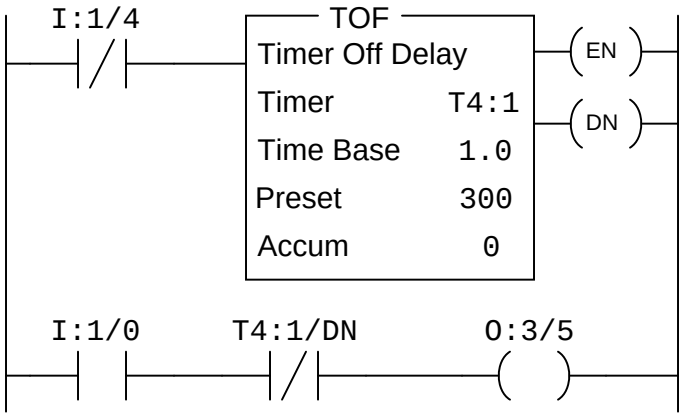
Question 85

Suppose we have an Allen-Bradley SLC 500 PLC with a water level switch and a temperature switch we need to connect to it:



The purpose of this PLC control is to start and stop a water pump drawing water from a well, to maintain a minimum water level in a storage tank. The level switch measures the water level in the storage tank to control the pump. The problem is, the pump will overheat if run continuously, so a high-temperature "cutout" switch is installed at the motor to sense motor temperature and shut off the pump if the motor gets too hot. The PLC will immediately shut off the motor if it senses a high temperature, and refuse to re-start the motor for at least 5 minutes after the temperature has fallen below the temperature switch's trip point.

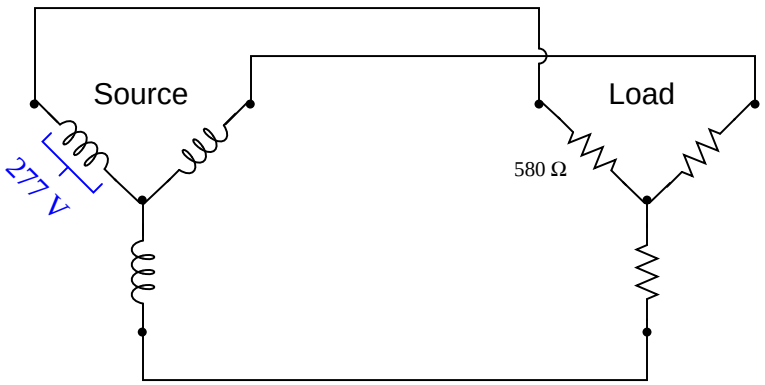
Someone else has already written the program for this PLC, leaving you to figure out which contact on each switch (NO or NC) must be connected to which terminal on the input card. Sketch wires for all connections to complete this system, based on this pre-written Ladder Diagram program:



file i02253

Question 86

Calculate all voltages, currents, and total power in this balanced Y-Y system:

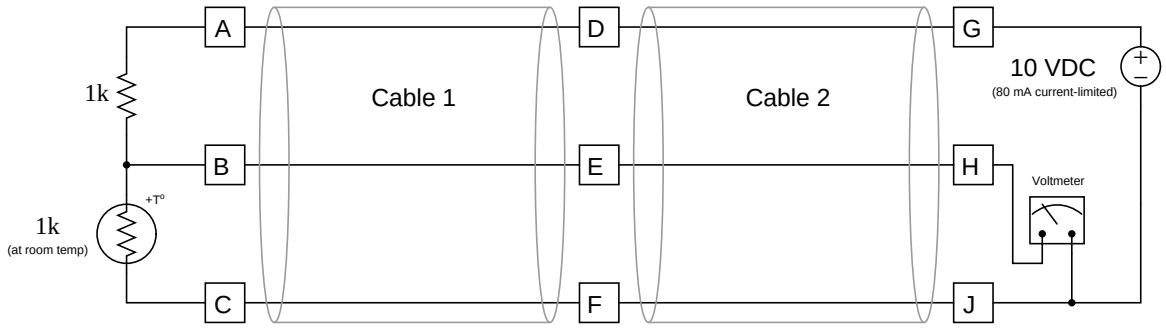


- E_{line} =
- I_{line} =
- $E_{phase(source)}$ =
- $I_{phase(source)}$ =
- $E_{phase(load)}$ =
- $I_{phase(load)}$ =
- P_{total} =

file i02421

Question 87

The following circuit senses temperature using a thermistor with a positive temperature coefficient (i.e. resistance increases as temperature increases):



First, determine the voltage we should read at the voltmeter with the thermistor at or near room temperature.

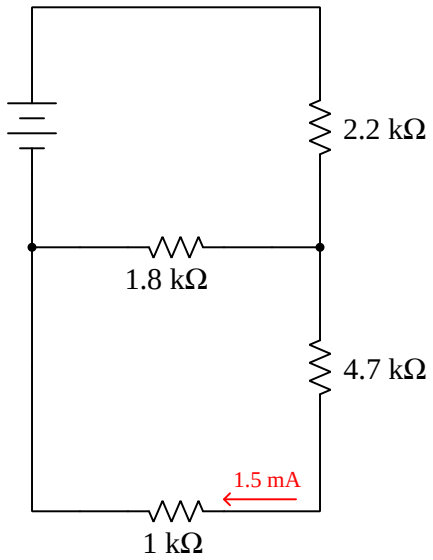
Next, identify the likelihood of each specified fault for this circuit, supposing the voltmeter registers 0 volts with the thermistor at room temperature, and a voltage measurement taken between terminals D and F registers 10 volts. Consider each fault one at a time (i.e. no coincidental faults), determining whether or not each fault could independently account for *all* measurements and symptoms in this circuit.

Fault	**Possible**	**Impossible**
Thermistor failed open		
Fixed resistor failed open		
Wire A-D failed open		
Wire F-J failed open		
Wire E-H failed open		
Thermistor failed shorted		
Fixed resistor failed shorted		
Short between terminals G-H		
Short between terminals E-F		
Short between terminals D-E		

file i02924

Question 88

Determine all component voltages and currents in this circuit, being sure to mark directions of all currents (conventional flow notation) and polarities of all voltages:



file i02525

Question 89

Suppose a single-phase AC load draws a current of 16.5 amps at 237 volts (RMS). If the measured power factor of this load is 0.85, calculate the *true power* (P) dissipated by the load as well as its *apparent power* (S). Be sure to include the proper unit of measurement (e.g. VA, VAR, or W) with each answer!

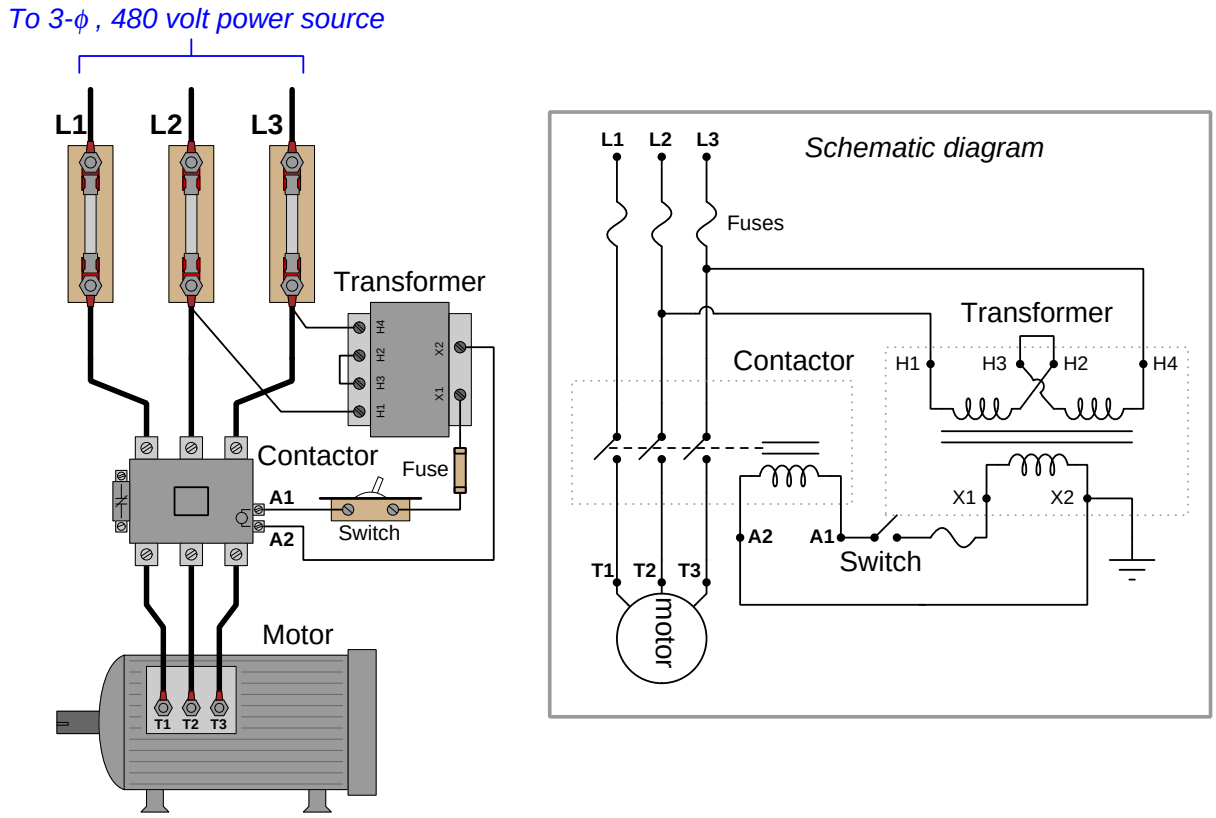
$P =$

$S =$

file i02422

Question 90

In this 480 volt AC induction motor control circuit (sometimes referred to as a "bucket"), a three-pole relay (typically called a *contactor*) is used to switch power on and off to the motor. The contactor itself is controlled by a smaller switch, which receives 120 volts AC from a step-down transformer to energize the contactor's magnetic coil. Although this motor control circuit used to work just fine, today the motor refuses to start.



Using your AC voltmeter, you measure 476 volts AC between L1 and L2, 477 volts AC between L2 and L3, and 475 volts AC between L1 and L3. You also measure 477 volts between transformer terminals H1 and H4. With the switch in the "on" position, you measure 0.5 volts AC between terminals X1 and X2 on the transformer. From this information, identify the following:

- Two components or wires in the circuit that you know cannot be failed either open or shorted, besides the 480 volt AC source which is obviously operational.

- Two different component or wire failures in the circuit, either one of which could account for the problem and all measured values, and the types of failures they would be (either open or shorted).

file i03174

Question 91

Lab Exercise – introduction

Your team's task is to construct a system controlled by a PLC. The system you choose to build shall use (at minimum) discrete input(s), discrete output(s), and either counter or timer functions. This system will be expanded during the next course to include a three-pole contactor, so designing the system with this in mind (or simply installing the contactor in this exercise) will save you time later. Project ideas include:

- Air compressor control, with high and low air pressure switches
- Water sump pump control, with high and low water level switches
- Other alternatives? *Must be pre-approved by instructor!*

In addition to functioning properly, the PLC program must be *fully* documented and edited for cleanliness and good programming form. This includes labels (aliases, or symbolic names) for all inputs and outputs, and comments for each and every rung of logic explaining the rungs' functions. Although there will be only one program submitted by each team, completion of this objective is individual, with each student explaining (at least) a part of the PLC program to the instructor.

Objective completion table:

Performance objective	Grading	1	2	3	4	Team
Team meeting and prototype sketch (do *first!*)	mastery	–	–	–	–	
Circuit design challenge	mastery					– – – –
Complete I/O list	mastery	–	–	–	–	
Prototype PLC program (*before programming!*)	mastery	–	–	–	–	
Final wiring diagram and system inspection	mastery					– – – –
Demonstration of working system	mastery	–	–	–	–	
Final PLC program inspection	mastery					– – – –
Lab question: Wiring connections	proportional					– – – –
Lab question: Commissioning	proportional					– – – –
Lab question: Mental math	proportional					– – – –
Lab question: Diagnostics	proportional					– – – –

The only "proportional" scoring in this activity are the lab questions, which are answered by each student individually. A listing of potential lab questions are shown at the end of this worksheet question. The lab questions are intended to guide your labwork as much as they are intended to measure your comprehension, and as such the instructor may ask these questions of your team day by day, rather than all at once (on a single day).

It is essential that your team plans ahead what to accomplish each day. A short (10 minute) team meeting at the beginning of each lab session is a good way to do this, reviewing what's already been done, what's left to do, and what assessments you should be ready for. There is a lot of work involved with building, documenting, and troubleshooting these working instrument systems!

As you and your team work on this system, you will invariably encounter problems. You should always attempt to solve these problems as a team before requesting instructor assistance. If you still require instructor assistance, write your team's color on the lab whiteboard with a brief description of what you need help on. The instructor will meet with each team in order they appear on the whiteboard to address these problems.

Lab Exercise – team meeting and prototype sketch

An important first step in completing this lab exercise is to **meet with your instructor** as a team to discuss safety concerns, team performance, and specific roles for team members. If you would like to emphasize exposure to certain equipment (e.g. use a particular type of control system, certain power tools), techniques (e.g. fabrication), or tasks to improve your skill set, this is the time to make requests of your team so that your learning during this project will be maximized.

An absolutely essential step in completing this lab exercise is to work together as a team to **sketch a prototype diagram** showing what you intend to build. This usually takes the form of a simple electrical schematic and/or loop diagram showing all electrical connections between components, as well as any tubing or piping for fluids. This prototype sketch need not be exhaustive in detail, but it does need to show enough detail for the instructor to determine if all components will be correctly connected for their safe function.

For example, if you intend to connect field devices to a PLC (Programmable Logic Controller), your prototype sketch must show how those devices will connect to typical input/output terminals on the PLC, where electrical power will be supplied, etc. Prototype sketches need not show all intermediary connections between components, such as terminal blocks in junction boxes between the field device and the controller.

You should practice good problem-solving techniques when creating your prototype sketch, such as consulting equipment manuals for information on component functions and marking directions of electric current, voltage polarities, and identifying electrical sources/loads. Use this task as an opportunity to strengthen your analytical skills! Remember that you will be challenged in this program to do all of this on your own (during "capstone" assessments), so do not make the mistake of relying on your teammates to figure this out for you – instead, treat this as a problem *you* must solve and compare your results with those of your teammates.

Your team's prototype sketch is so important that the instructor will demand you provide this plan before any construction on your team's working system begins. *Any team found constructing their system without a verified plan will be ordered to cease construction and not resume until a prototype plan has been drafted and approved!* Similarly, you should not deviate from the prototype design without instructor approval, to ensure nothing will be done to harm equipment by way of incorrect connections. Each member on the team should have ready access to this plan (ideally possessing their own copy of the plan) throughout the construction process. Prototype design sketching is a skill and a habit you should cultivate in school and take with you in your new career.

Select a PLC with modular (add-on) I/O cards to provide sufficient complexity for the project. Monolithic "brick" PLCs (with no add-on I/O modules) are not acceptable for this project. An Allen-Bradley SLC 500 PLC would be a good choice, as well as a Siemens S7 series or an AutomationDirect Productivity 3000.

You will also need to select appropriate field devices (switches, pumps, etc.) for your project. You are free to use the field devices left over from the relay-based motor control lab if you prefer.

The next step should be finding appropriate documentation for your PLC. All PLC manufacturers provide manuals and datasheets for their products online. Use this documentation to identify how to properly wire, power, and program your team's PLC.

PLC equipment manuals always provide sample diagrams showing how external components may connect to the I/O points. Feel free to use these sample diagrams as templates for your prototype sketch. *This is the most challenging portion of your wiring, so be sure to work with your teammates to get this right!*

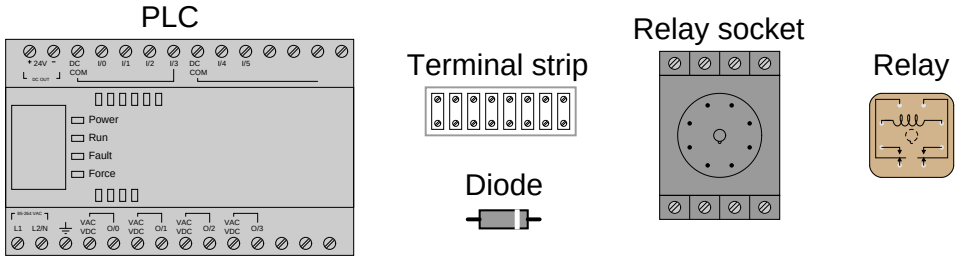
Planning a functioning system should take no more than a couple of hours if the team is working efficiently, and will save you hours of frustration (and possible component destruction!).

Lab Exercise – circuit design challenge

Connect an "ice cube" relay to one of the outputs on a PLC, so that the PLC can control the energization of the relay. All electrical connections must be made using a terminal strip (no twisted wires, crimp splices, wire nuts, spring clips, or "alligator" clips permitted). Program this PLC to implement a motor start/stop (latching) control function. *In order to ensure your program has not been pre-written in your computer prior to this assessment, you will be asked to sketch a correct ladder-diagram PLC program on paper to implement this function prior to using a computer.*

You must connect a "commutating" diode in parallel with the relay's coil to prevent the phenomenon known as "inductive kickback," which may otherwise damage the transistor output on a PLC. Note that incorrectly connecting this diode will present a short-circuit to the PLC, so you *must* get it right!

This exercise tests your ability to properly interpret the "pinout" of an electromechanical relay, properly wire a PLC output channel to control a relay's coil, properly polarize a commutating diode to prevent inductive kickback from damaging the PLC output, and use a terminal strip to organize all electrical connections. It also tests your ability to program motor start/stop logic using either a seal-in contact or latching (retentive) coil instructions.



The following components and materials will be available to you: assorted "ice cube" **relays** with DC-rated coils and matching **sockets** ; **terminal strips** ; 1N400X rectifying **diodes** ; lengths of **hook-up wire**. You will be expected to supply your own screwdrivers and multimeter for assembling and testing the circuit at your desk.

"Start" switch to input: ____ **"Stop" switch to input**: ____ **Relay to output**: ____

PLC program (instructor chooses): ____ Seal-in contact ____ Retentive coils

Lab Exercise – developing a PLC I/O list

It is a good idea when programming any computer system to first identify all the input and output signals to the system, as well as internal variables if possible, before commencing on the development of the program itself. In order to reinforce this practice, your team will be required to develop a complete list of all input and output points on your proposed system along with any tagnames (also known as "symbols" or "nicknames") identifying the function of each.

Once this list is complete and you are ready to begin developing the PLC program, you can enter all the tagnames and define the I/O points as your very first programming step. With this data in place, the writing of your program will be made easier because each I/O tag you reference will already be defined and labeled, reminding you of their functions within the system.

Here is a sample I/O list for a motor control PLC program:

Hardware I/O terminal	I/O type	Tagname	Notes
Card 1, terminal IN0	24 VDC discrete input	START_PB	Black pushbutton, momentary NO contacts
Card 1, terminal IN1	24 VDC discrete input	STOP_PB	Red pushbutton, momentary NC contacts
Card 1, terminal IN2	24 VDC discrete input	E_STOP	Red pushbutton, latching NC contacts
Card 2, terminal IN0	4-20 mA analog input	MTR_TEMP	Current signal scaled 0 to 150 deg F
Card 3, terminal OUT0	120 VAC discrete output	CONTACTOR	To terminal A1

Lab Exercise – wiring the system

The Instrumentation lab is set up to facilitate the construction of working systems, with over a dozen junction boxes, pre-pulled signal cables, and "racks" set up with 2-inch vertical pipes for mounting instruments. The only wires you should need to install to build a working system are those connecting the field instrument to the nearest junction box, and then small "jumper" cables connecting different pre-installed cables together within intermediate junction boxes.

Your team's PLC must be installed in a suitable electrical enclosure, with AC power fed to it through a fuse or circuit breaker (on the "hot" conductor only), and firmly grounded (the ground conductor of the power cord securely fastened to the metal frame of the enclosure and the PLC chassis).

All I/O wiring should be neatly loomed together and/or run through wire duct ("Panduit"). Power to I/O cards must be routed through their own fuses so that I/O power may be disconnected independently of power to the PLC processor and rack.

Common mistakes:

- Neglecting to consult the manufacturer's documentation for field instruments (e.g. how to wire them, how to calibrate them).
- Proceeding with wiring *before* creating an initial sketch of the circuitry and checking that sketch for errors.
- Mounting the field instrument(s) in awkward positions, making it difficult to reach connection terminals or to remove covers when installed.
- Failing to tug on each and every wire where it terminates to ensure a mechanically sound connection.
- Students working on portions of the system in isolation, not sharing with their teammates what they did and how. It is important that the whole team learns all aspects of their system!

Building a functioning system should take no more than one full lab session (3 hours) if all components are readily available and the team is working efficiently!

Lab Exercise – programming the system

Like wiring a control system, programming one is best done with thoughtful planning rather than a "design-as-you-build" approach. Each team will work with the instructor to develop a "prototype" PLC program, usually on a whiteboard or on paper. *Having multiple teams prototype their programs on whiteboards within the same classroom works well to foster peer review of programming, where teams analyze and critique other teams' programming solutions!* Your prototype program should completely address the following points:

- Identify all inputs to the PLC, giving each one a sensible tagname
- Identify all signal outputs from the PLC, giving each one a sensible tagname
- Identify all major program functions (i.e. *What must this program do?*)
- Identify all internal variables necessary for these functions, giving each one a sensible tagname
- Identify all system variables necessary for these functions (e.g. real-time clock/calendar variables)

The importance of identifying and naming all relevant variables is paramount to "clean" programming. This is especially true when an HMI (Human-Machine Interface) is to be connected to the PLC, and all relevant variables must be named there as well.

A reasonable approach to developing a robust program prototype is to create your prototype in your own personal ("brick") PLC, de-bugging it there with all the switches in place to simulate input signals. Even if your personal PLC is a different model (or manufacture) than the project PLC, this is a very helpful exercise. Furthermore, it allows you to continue program development outside of school when you do not have access to the project PLC.

Only after a prototype program is developed should you begin programming the project PLC. I recommend the following steps:

- Establish communications between PLC and personal computer (PC)
- Connect all I/O cards (modules) to the PLC and get them recognized by the processor
- Assign tagnames to all relevant variables, beginning with I/O points
- Enter a simplified version of the program, running to check for "bugs"
- Diagnose any program problems
- Add complexity to the program (e.g. additional features) and run to check for "bugs"
- Repeat last two steps as often as necessary
- Add comments to each and every line of the program, explaining how it functions

The final program should be well-documented, clean, and as simple as possible. All members of the team should have a hand in designing the program, and everyone *must* thoroughly understand how it works.

Common mistakes:

- Waiting too long after writing the program code to insert comments. This is best done immediately, while everything makes sense and is fresh in your memory!
- Insufficient commenting – only makes sense to the person who did the programming
- Students working on portions of the program in isolation, not sharing with their teammates what they did and how. It is important that the whole team learns all aspects of their system!

Lab Exercise – documenting the system

Each student must sketch their own *wiring diagram* for their team's system, following industry-standard conventions. Sample diagrams for input and output wiring are shown in the next question in this worksheet. These wiring diagrams must be *comprehensive* and *detailed*, showing every connection, every cable, every terminal block, etc. The principle to keep in mind here is to make the wiring diagram so complete and unambiguous that anyone can follow it to see what connects to what, even someone unfamiliar with industrial instrumentation. In industry, control systems are often constructed by contract personnel with limited understanding of how the system is supposed to function. The associated diagrams they follow must be so complete that they will be able to connect everything properly without necessarily understanding how it is supposed to work.

When your entire team is finished drafting your individual wiring diagrams, call the instructor to do an inspection of the system. Here, the instructor will have students take turns going through the entire system, with the other students checking their diagrams for errors and omissions along the way. During this time the instructor will also inspect the quality of the installation, identifying problems such as frayed wires, improperly crimped terminals, poor cable routing, missing labels, lack of wire duct covers, etc. The team must correct all identified errors in order to receive credit for their system.

After successfully passing the inspection, each team member needs to place their wiring diagram in the diagram holder located in the middle of the lab behind the main control panel. When it comes time to troubleshoot another team's system, this is where you will go to find a wiring diagram for that system!

Common mistakes:

- Forgetting to label all signal wires (see example wiring diagrams).
- Forgetting to label all field instruments with their own tag names (e.g. PSL-83).
- Forgetting to note all wire colors.
- Forgetting to put your name on the wiring diagram!
- Basing your diagram off of a team-mate's diagram, rather than closely inspecting the system for yourself.

Creating and inspecting accurate wiring diagrams should take no more than one full lab session (3 hours) if the team is working efficiently!

Lab questions

- **Wiring connections**
- Determine correct wire connections between field components and a PLC I/O card to create a working PLC input or output circuit, based on diagrams of components with terminals labeled
- Correctly determine all electrical sources and loads, as well as all voltage polarities and current directions, in a DC input or output circuit, based on diagrams of field components and the PLC's I/O card with terminals labeled

- **Commissioning and Documentation**
- Explain what is meant by the term "sinking" with regard to a PLC input card (DC)
- Explain what is meant by the term "sourcing" with regard to a PLC input card (DC)
- Explain what is meant by the term "sinking" with regard to a PLC output card (DC)
- Explain what is meant by the term "sourcing" with regard to a PLC output card (DC)
- Explain what a "TRIAC" PLC output card is, and how it differs from DC output cards
- Explain what a "relay" PLC output card is, and how it differs from sourcing or sinking DC output cards
- Explain the distinction between "online" and "offline" programming modes for a PLC

- **Mental math** (no calculator allowed!)
- Convert a binary number into decimal
- Convert a binary number into hexadecimal
- Convert a decimal number into binary
- Convert a hexadecimal number into binary
- Convert a hexadecimal number into decimal

- **Diagnostics**
- Examine a PLC program and identify any mistakes in it
- Determine whether or not a given diagnostic test will provide useful information, given a set of symptoms exhibited by a failed system
- Identify at least two plausible faults given the results of a diagnostic test and a set of symptoms exhibited by a failed system
- Propose a diagnostic test for troubleshooting a failed system and then explain the meanings of two different test results

Wiring diagram requirements

- **Wiring diagram**
- Proper symbols and designations used for all components.
- Relay coil and contacts properly named.

- **Text descriptions**
- Each instrument documented below (tag number, description, etc.).
- Calibration (input and output ranges) given for each instrument, as applicable.

- **Connection points**
- All terminal blocks properly labeled.
- All terminals shown in proper order on diagram.
- All I/O cards and points fully labeled (complete with program addresses).
- All wires are numbered.
- All electrically-common points in the circuit shall bear the same wire number.
- All wire colors shown next to each terminal.

- **Cables and tubes**
- Multi-pair cables or pneumatic tube bundles going between junction boxes and/or panels need to have unique numbers (e.g. "Cable 10") as well as numbers for each pair (e.g. "Pair 1," "Pair 2," etc.).

- **Energy sources**
- All power source intensities labeled (e.g. "24 VDC," "120 VAC," "20 PSI")
- All shutoff points labeled (e.g. "Breaker #5," "Valve #7")

file i03655

Question 92

Wiring diagram requirements

Perhaps the most important rule to follow when drafting a wiring diagram is *your diagram should be complete and detailed enough that even someone who is not a technician could understand where every wire should connect in the system!*

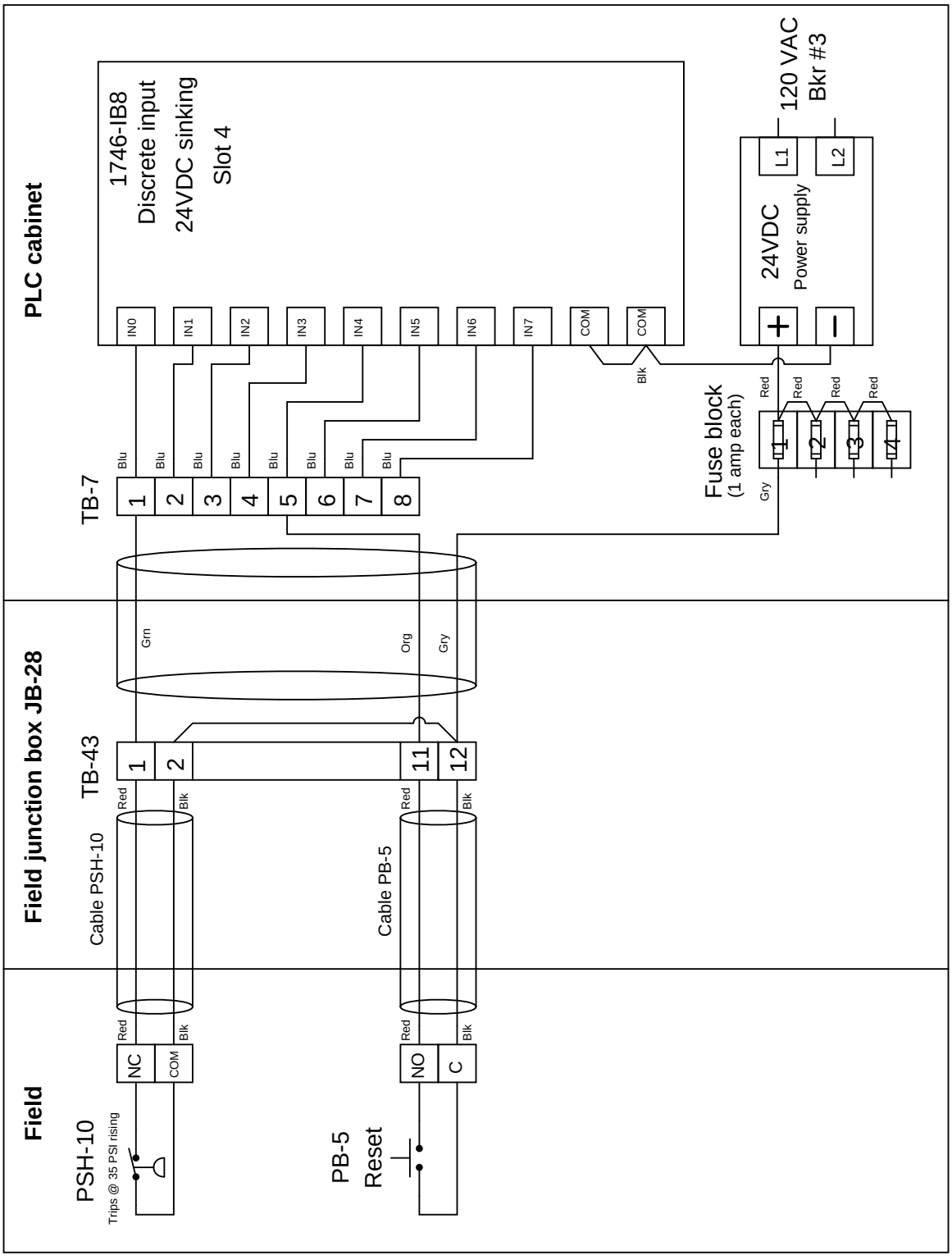
- **Field device symbols**
- Proper electrical symbols and designations used for all field devices.
- *Optional:* Trip settings written next to each process switch.

- **PLC I/O cards**
- All terminals labeled, even if unused in your system.
- Model number, I/O type, and PLC slot number should be shown for each and every card.

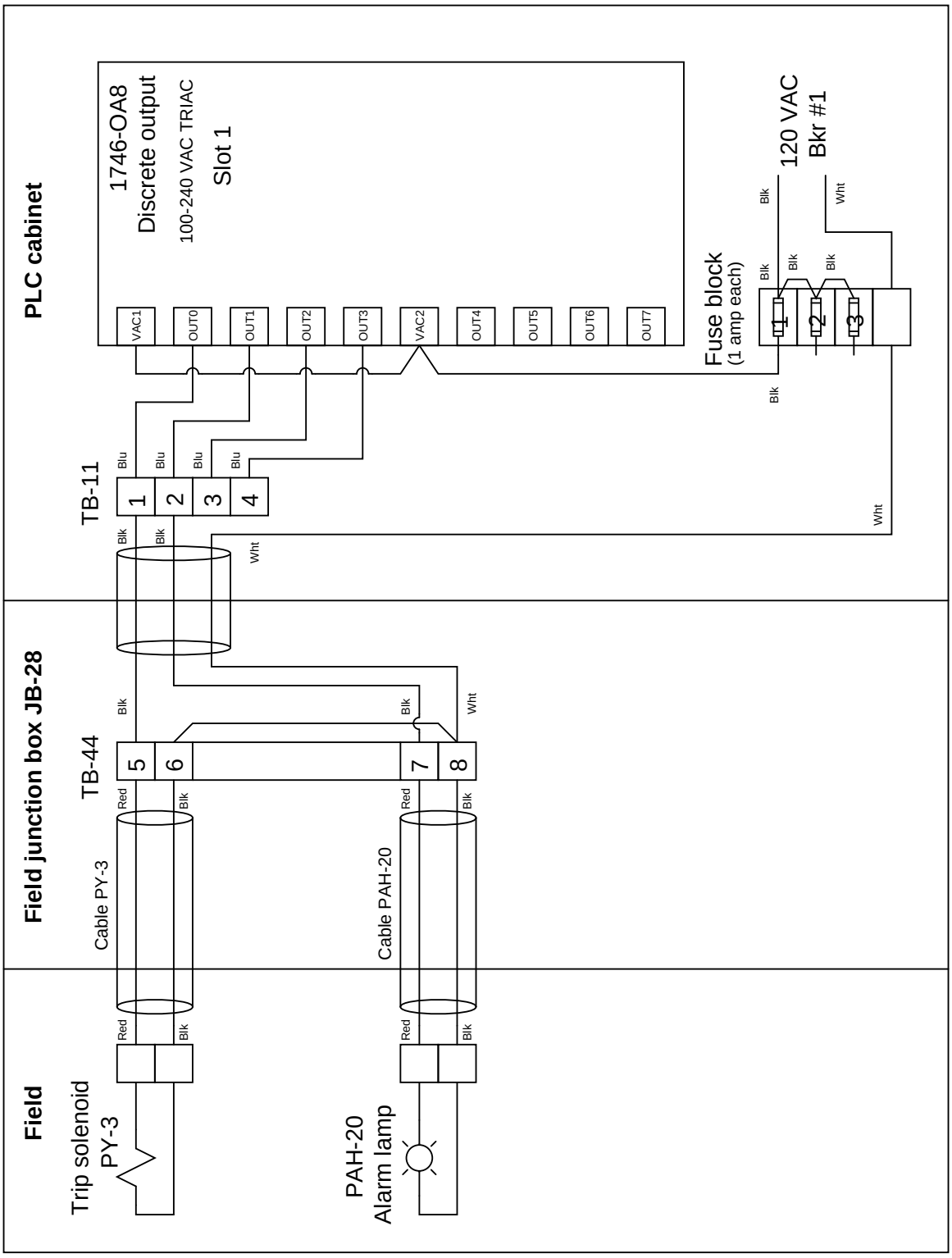
- **Connection points**
- All terminals properly labeled.
- All terminal blocks properly labeled.
- All junction ("field") boxes shown as distinct sections of the loop diagram, and properly labeled.
- All control panels shown as distinct sections of the loop diagram, and properly labeled.
- All wire colors shown next to each terminal.
- All terminals on devices labeled as they appear on the device (so that anyone reading the diagram will know which device terminal each wire goes to).

- **Energy sources**
- All power source intensities labeled (e.g. "24 VDC," "120 VAC," "480 VAC 3-phase")
- All shutoff points labeled (e.g. "Breaker #5," "Valve #7")

Sample Input Wiring Diagram



Sample Output Wiring Diagram



file i01880

Answer 1

Answer 2

Input register, byte 1, bit 4: `I1.4`

Output register, byte 0, bit 2: `Q0.2`

Variable memory double word, starting at byte 105: `VD105` (a double-word consisting of 4 bytes, or 32 bits)

Answer 3

Input file, element 1, bit 4: `I:1/4`

Output file, element 0, bit 2: `O:0/2`

Timer 6 accumulator word: `T4:6.ACC`

Answer 4

For the Allen-Bradley MicroLogix example, the lamp will energize only when switch 0 is turned off and switch 1 is turned on.

For the Siemens S7-200 example, the lamp will energize when switch 0 is turned on or if switch 1 is turned off, or both conditions occur simultaneously.

For the Koyo example, the lamp will energize according to the *Exclusive-OR* function with switch 1 and switch 2. The lamp energizes when switch 1 is on and switch 2 is off, or when switch 1 is off and switch 2 is on.

Answer 5

Answer 6

Answer 7

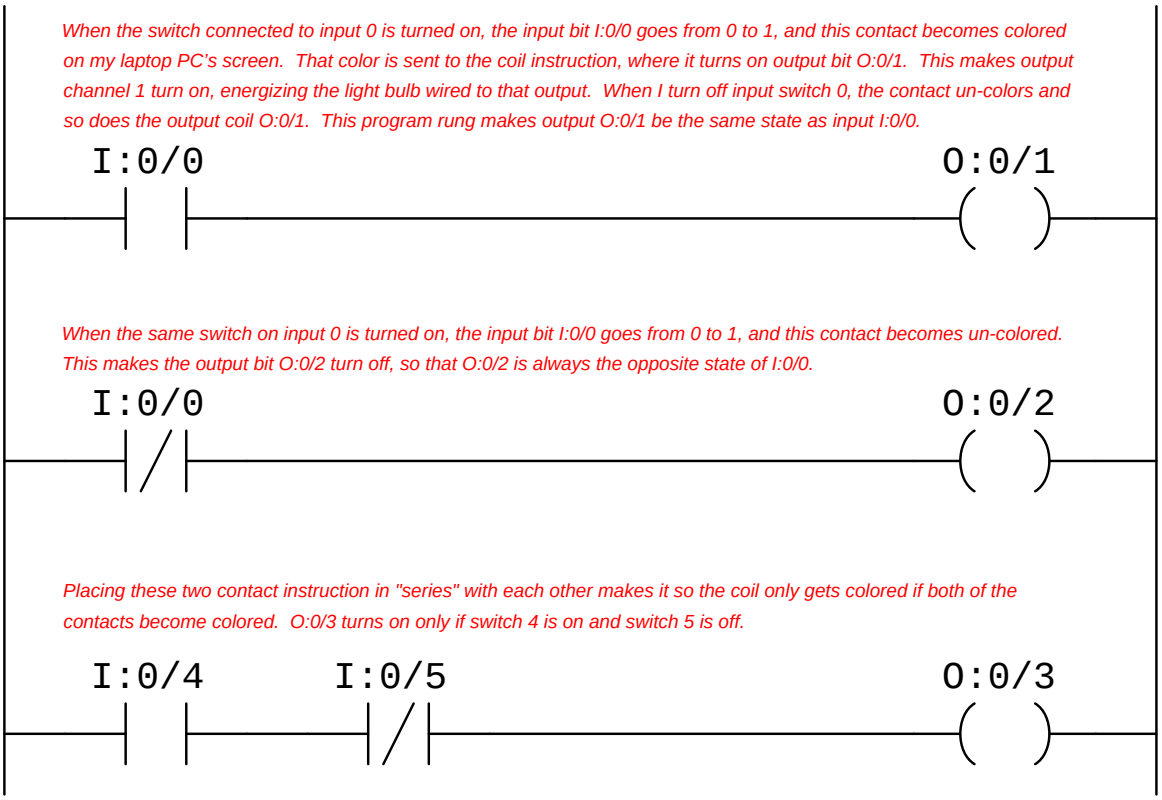
Answer 8

A good problem-solving technique to apply in both diagrams is *annotation*, where you indicate the presence of continuity and power versus non-continuity/unpowered. In PLC programs this usually appears in the form of color-highlighting surrounding each instruction symbol (virtual contact or virtual coil).

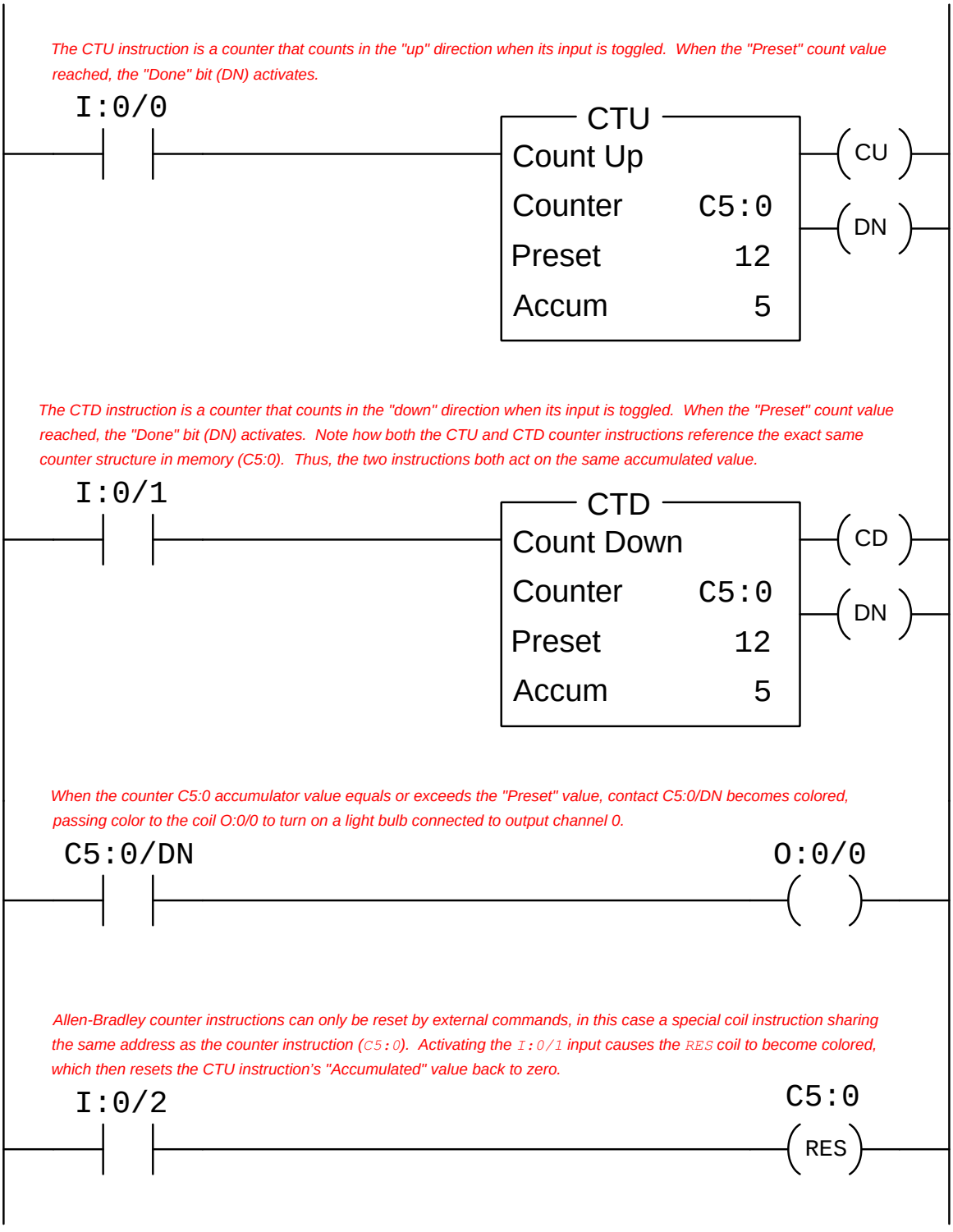
Answer 9

Answer 10

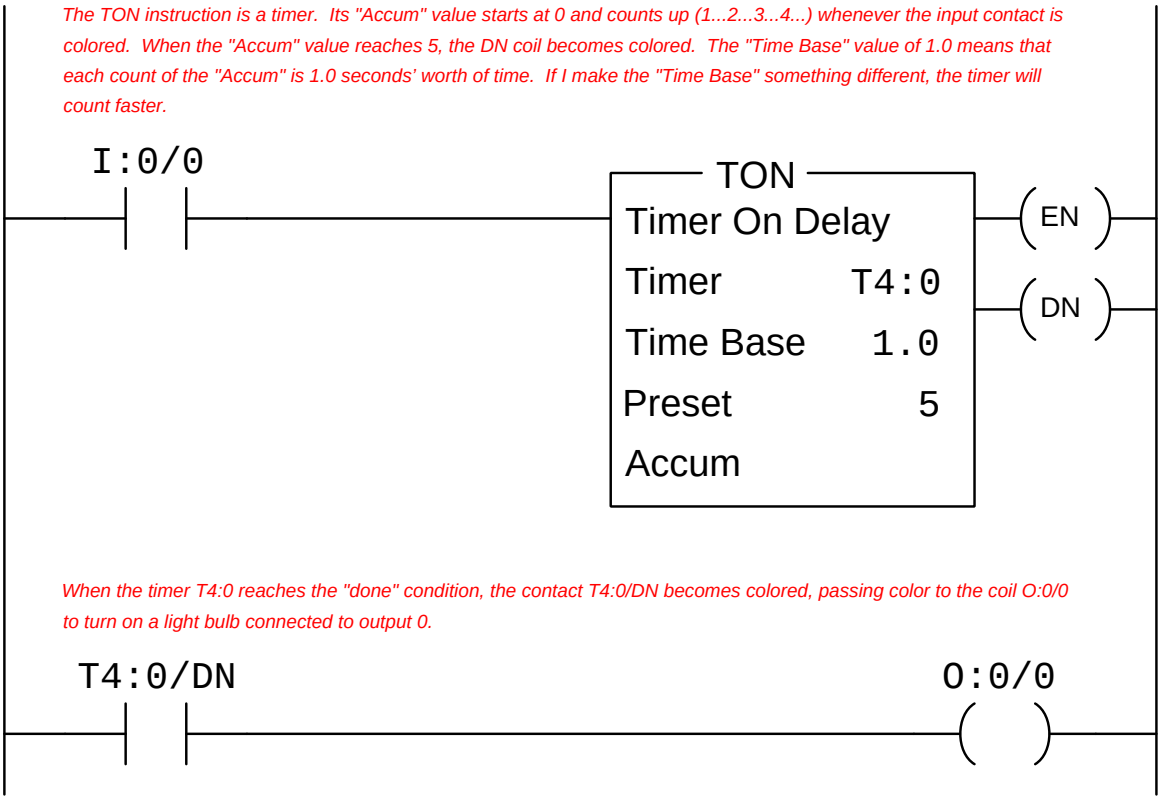
Demonstration program showing some basic bit instructions in an Allen-Bradley MicroLogix PLC:


When the switch connected to input 0 is turned on, the input bit I:0/0 goes from 0 to 1, and this contact becomes colored on my laptop PC's screen. That color is sent to the coil instruction, where it turns on output bit O:0/1. This makes output channel 1 turn on, energizing the light bulb wired to that output. When I turn off input switch 0, the contact un-colors and so does the output coil O:0/1. This program rung makes output O:0/1 be the same state as input I:0/0.
I:0/0
O:0/1
When the same switch on input 0 is turned on, the input bit I:0/0 goes from 0 to 1, and this contact becomes un-colored. This makes the output bit O:0/2 turn off, so that O:0/2 is always the opposite state of I:0/0.
I:0/0
O:0/2
Placing these two contact instruction in "series" with each other makes it so the coil only gets colored if both of the contacts become colored. O:0/3 turns on only if switch 4 is on and switch 5 is off.
I:0/4
I:0/5
O:0/3

Demonstration program showing "up" and "down" counter instructions in an Allen-Bradley MicroLogix PLC:


The CTU instruction is a counter that counts in the "up" direction when its input is toggled. When the "Preset" count value reached, the "Done" bit (DN) activates.
I:0/0
CTU
Count Up
Counter C5:0
Preset 12
Accum 5
CU
DN
The CTD instruction is a counter that counts in the "down" direction when its input is toggled. When the "Preset" count value reached, the "Done" bit (DN) activates. Note how both the CTU and CTD counter instructions reference the exact same counter structure in memory (C5:0). Thus, the two instructions both act on the same accumulated value.
I:0/1
CTD
Count Down
Counter C5:0
Preset 12
Accum 5
CD
DN
When the counter C5:0 accumulator value equals or exceeds the "Preset" value, contact C5:0/DN becomes colored, passing color to the coil O:0/0 to turn on a light bulb connected to output channel 0.
C5:0/DN
O:0/0
Allen-Bradley counter instructions can only be reset by external commands, in this case a special coil instruction sharing the same address as the counter instruction (C5:0). Activating the I:0/1 input causes the RES coil to become colored, which then resets the CTU instruction's "Accumulated" value back to zero.
I:0/2
C5:0
RES

Demonstration program showing an on-delay timer instruction in an Allen-Bradley MicroLogix PLC:


The TON instruction is a timer. Its "Accum" value starts at 0 and counts up (1...2...3...4...) whenever the input contact is colored. When the "Accum" value reaches 5, the DN coil becomes colored. The "Time Base" value of 1.0 means that each count of the "Accum" is 1.0 seconds' worth of time. If I make the "Time Base" something different, the timer will count faster.
I:0/0
TON
Timer On Delay
Timer T4:0
Time Base 1.0
Preset 5
Accum
EN
DN
When the timer T4:0 reaches the "done" condition, the contact T4:0/DN becomes colored, passing color to the coil O:0/0 to turn on a light bulb connected to output 0.
T4:0/DN
O:0/0

Answer 11

Answer 12

Answer 13

Bit statuses:

- `I:0/0` = 1
- `I:0/1` = 0
- `I:0/2` = 1

Answer 14

Bit statuses:

- `I0.2` = 1
- `I1.1` = 0

Answer 15

$L > 3$ feet, $P > 37$ PSI, and $T < 88^{o}$F

Answer 16

- `I0.2` = 0
- `I0.5` = 1
- `I1.1` = 0
- `Q0.1` = 1
- `Q0.6` = 0

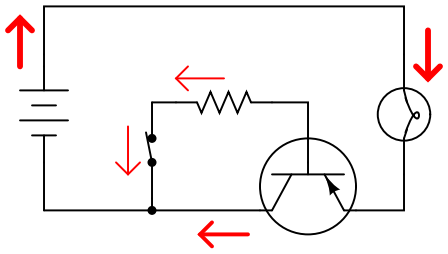
Answer 17

Switch statuses:

- Switch A = **released**
- Switch B = **pressed**
- Switch C = **released**

The lamp will be energized.

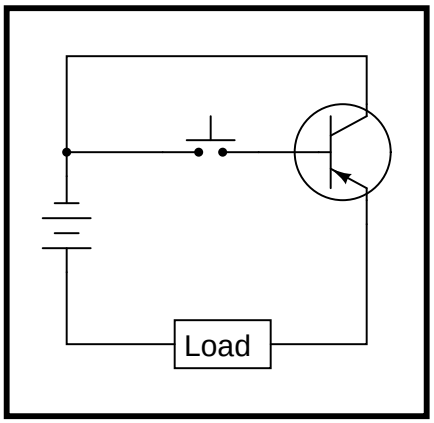
Answer 18



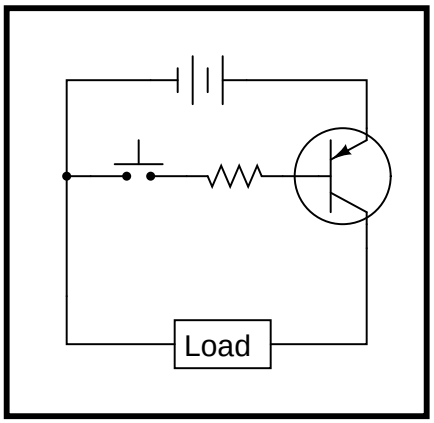
All currents shown using conventional flow notation

Answer 19

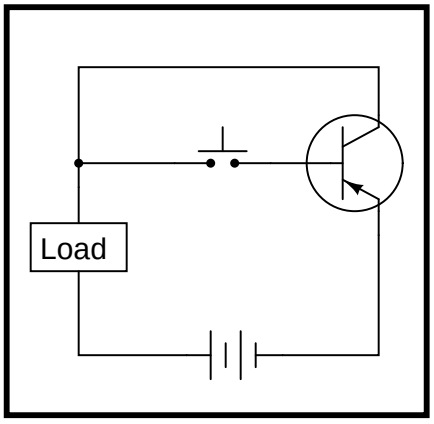
Circuit 1 ***This will work!***



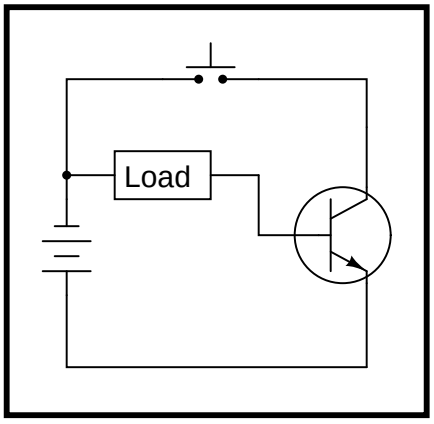
Circuit 2 ***This will work!***



Circuit 3 **This circuit is bad**

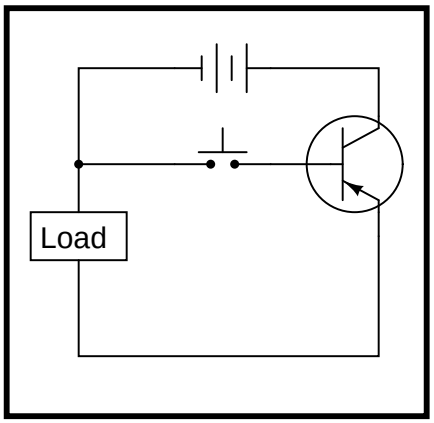


Circuit 4 **This circuit is bad**

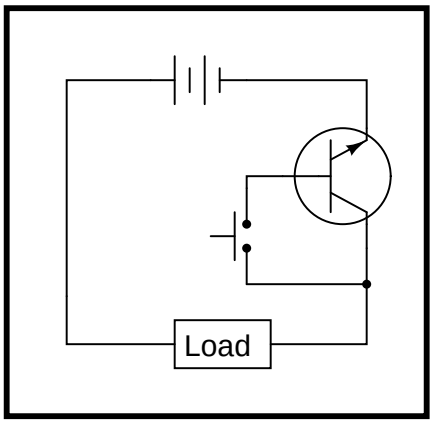


Answer 20

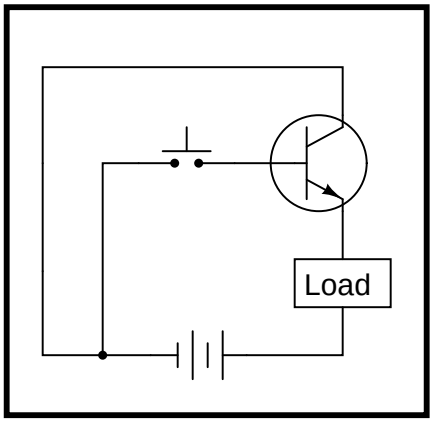
Circuit 1 **This circuit is bad**



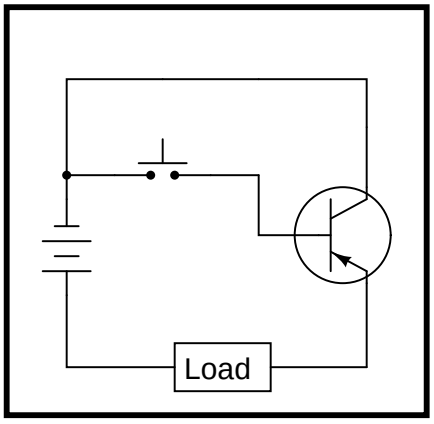
Circuit 2 ***This will work!***



Circuit 3 **This circuit is bad**

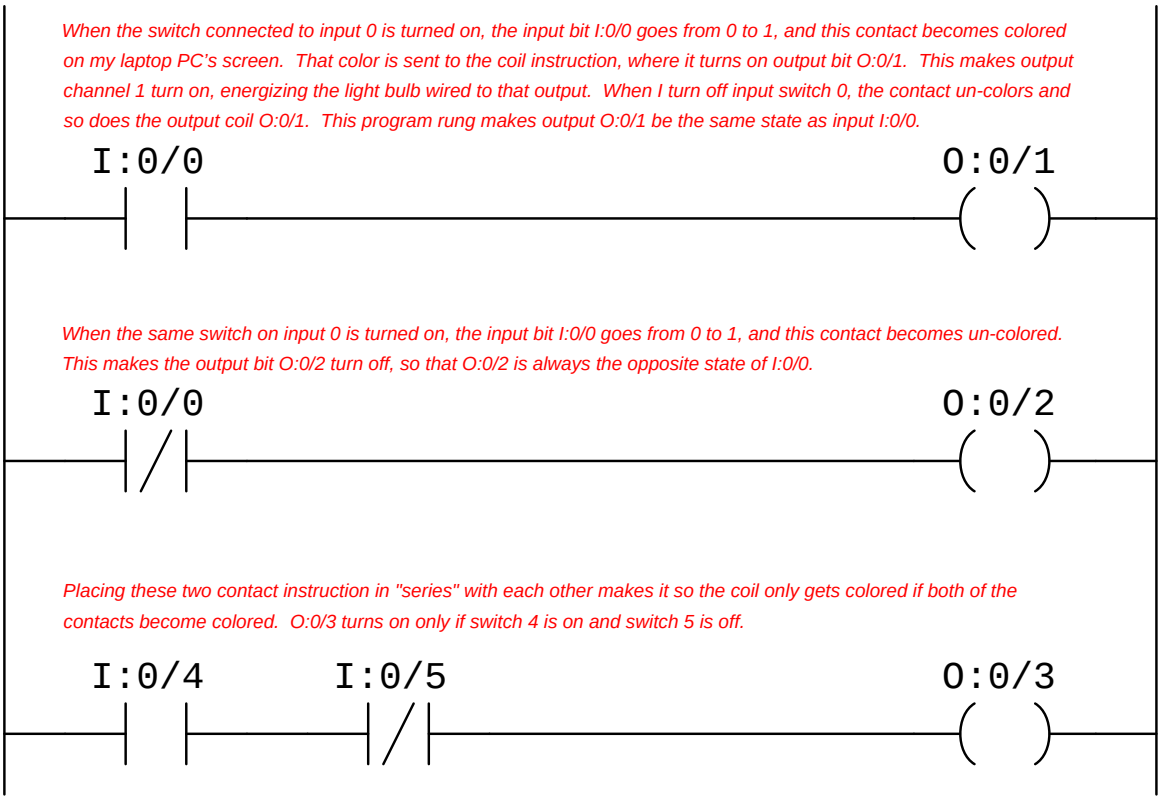


Circuit 4 ***This will work!***



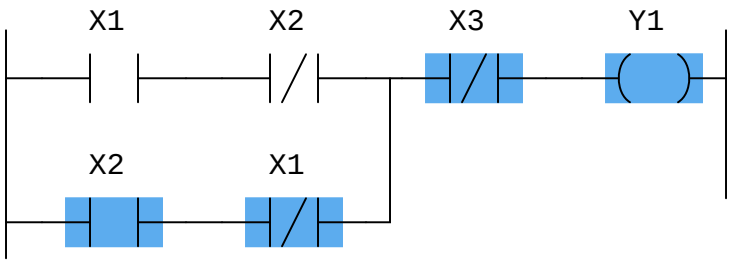
Answer 21

Demonstration program showing some basic bit instructions in an Allen-Bradley MicroLogix PLC:


When the switch connected to input 0 is turned on, the input bit I:0/0 goes from 0 to 1, and this contact becomes colored on my laptop PC's screen. That color is sent to the coil instruction, where it turns on output bit O:0/1. This makes output channel 1 turn on, energizing the light bulb wired to that output. When I turn off input switch 0, the contact un-colors and so does the output coil O:0/1. This program rung makes output O:0/1 be the same state as input I:0/0.
I:0/0
O:0/1
When the same switch on input 0 is turned on, the input bit I:0/0 goes from 0 to 1, and this contact becomes un-colored. This makes the output bit O:0/2 turn off, so that O:0/2 is always the opposite state of I:0/0.
I:0/0
O:0/2
Placing these two contact instruction in "series" with each other makes it so the coil only gets colored if both of the contacts become colored. O:0/3 turns on only if switch 4 is on and switch 5 is off.
I:0/4
I:0/5
O:0/3

Note: your own demonstration program should contain some *retentive* coil instruction as well, in order for you to be able to observe what these instructions do and how their operation differs from that of "regular" coil instructions!

Answer 22



- Y1 = 1

Answer 23

- I0.7 = 0
- I1.1 = 1
- Q0.1 = 0
- Q0.3 = 1

Answer 24

Neither output will activate to energize either lamp.

Answer 25

Here are just a couple of possible problems to account for what we are seeing. There are definitely more possible faults than what are listed here:

- Overload contact tripped (open)
- Wire connecting "Stop" switch to OL contact failed open

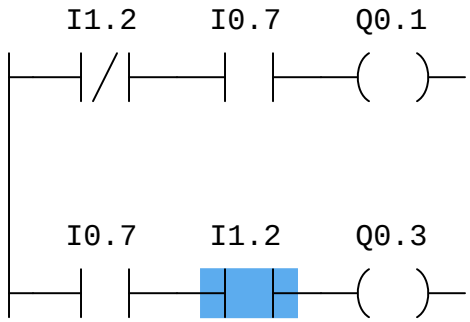
Answer 26

Answer 27

There are no answers provided here! For help, consult the "instruction set" reference manual for your PLC, which will describe in detail how each type of instruction is supposed to function in your PLC.

Answer 28

If both switches are pressed, switch A will be closed (I1.2 = 1) and switch B will be open (I0.7 = 0), leading to this condition of the program:



Neither output will activate, resulting in both lamps de-energized.

Answer 29

In order for the lamp to energize, virtual coil Y1 must be colored. In order to color this coil instruction, virtual contact X3 must be colored, and either virtual contacts X1 or *X2* must be colored. Since the X3 contact is NO and both X1 and X2 contacts are NC, this requires input X3 to be powered, and either input X1 or X2 to be unpowered.

Thus, NO pushbutton "C" must be pressed, and either NO pushbutton "A" released or NC pushbutton "B" pressed:

- Switch A = **released** *or* Switch B = **pressed**
- Switch C = **pressed**

Answer 30

Bit statuses:

- `I:0/0` = 0
- `I:0/1` = 0
- `I:0/3` = 1

Answer 31

Bit statuses:

- `I0.2` = 0
- `I1.1` = 0

Answer 32

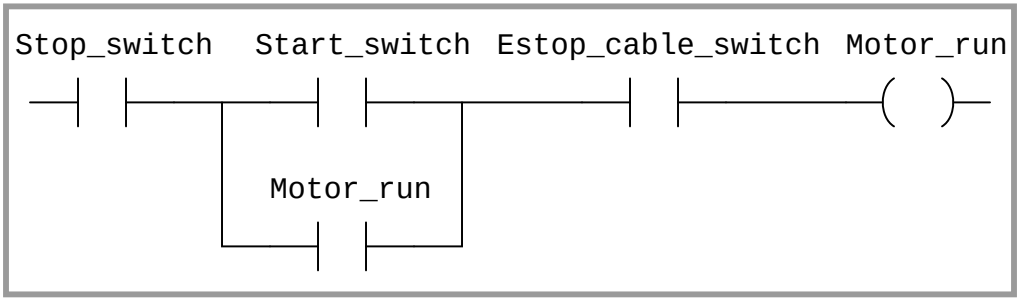
Bit statuses:

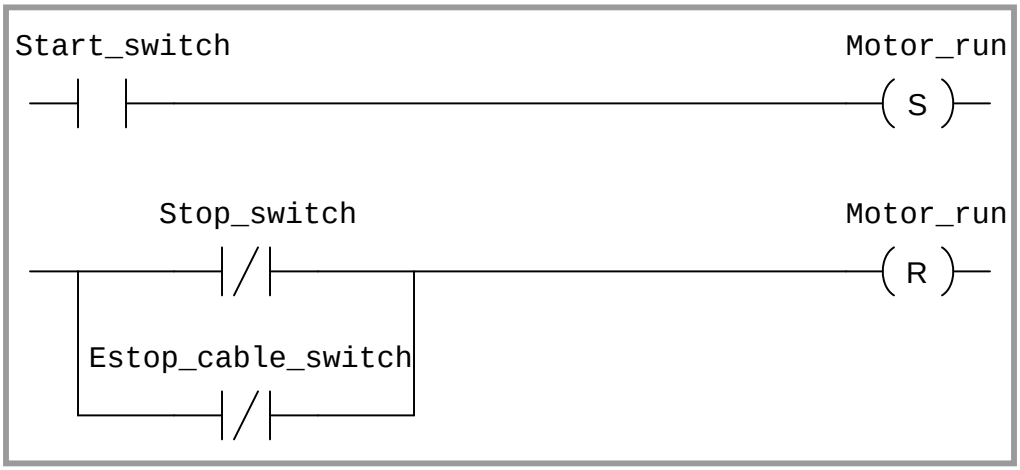
- `I:1/3` = 1
- `I:1/5` = 0

Answer 33

Answer 34

Two different program solutions:

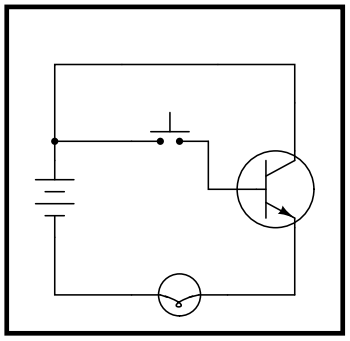




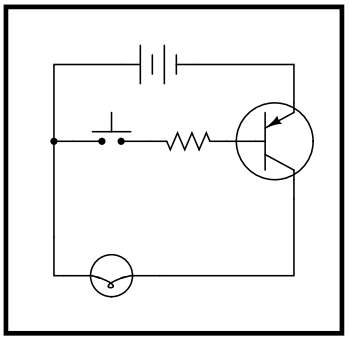
Answer 35

Remember that a bipolar transistor requires current through the base-emitter junction in order to turn on, and thereby let load current pass between collector and emitter.

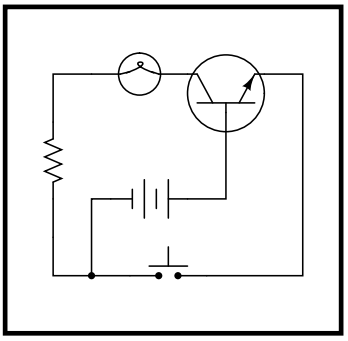
Circuit 1 ***This circuit will work!***



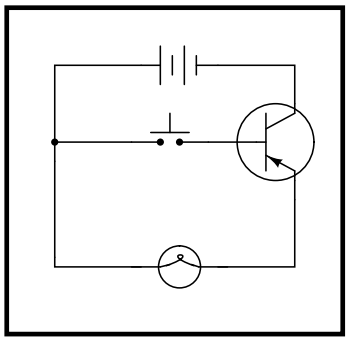
Circuit 2 **This circuit is bad**



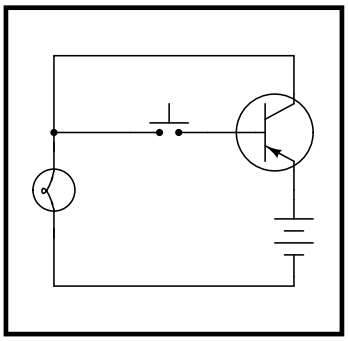
Circuit 3 **This circuit is bad**



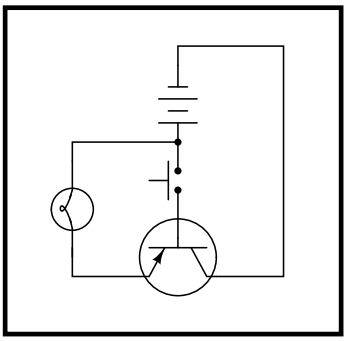
Circuit 4 **This circuit is bad**



Circuit 5 ***This circuit will work!***



Circuit 6 **This circuit is bad**



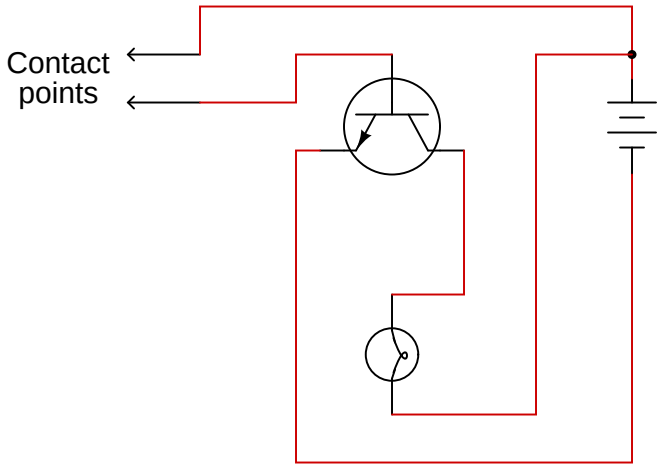
Circuit #3 is different from the other "bad" circuits. While the other bad circuits' lamps do not energize at all, the lamp in circuit #3 energizes weakly when the pushbutton switch is open (not actuated). This is due to the fact that lamp current will naturally pass through the base-collector PN junction as though it were a simple diode, regardless of the switch's state.

Answer 36

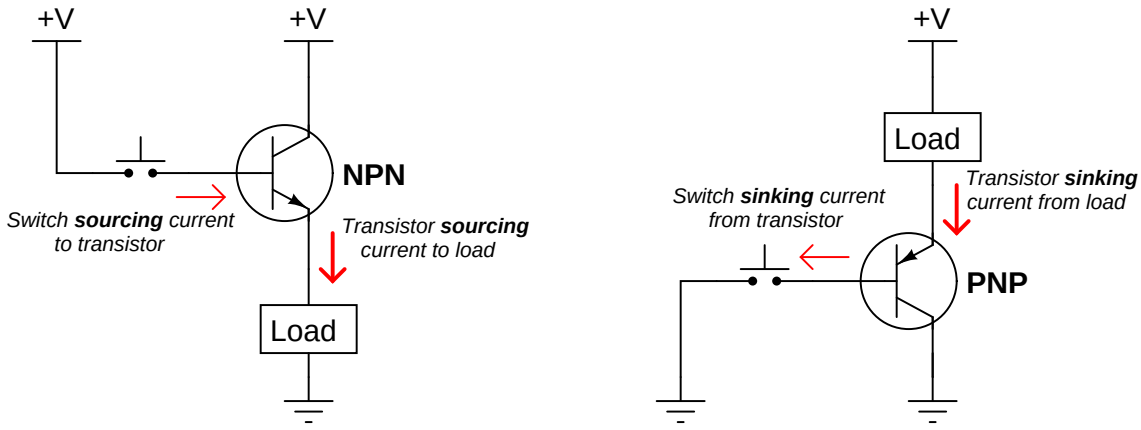
Circuits 3, 5, and 6 are flawed, because the emitter-base junctions of their transistors are overpowered every time the switch closes.

Hint: draw the respective paths of switch and lamp current for each circuit!

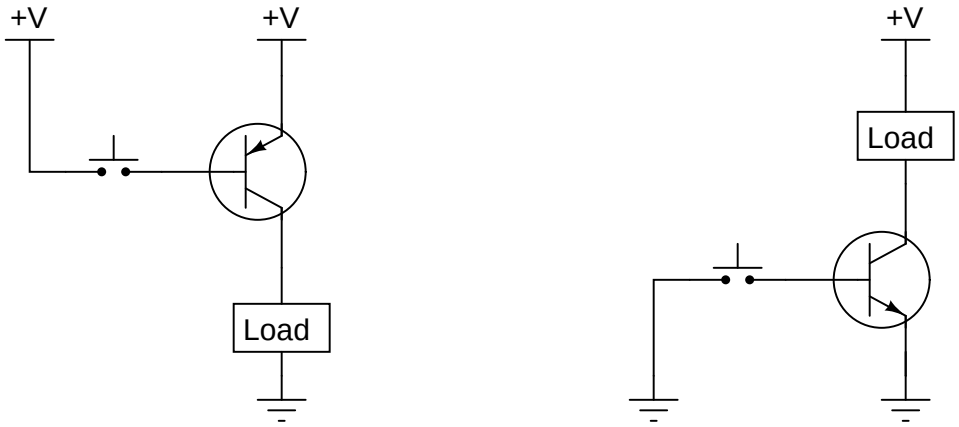
Answer 37



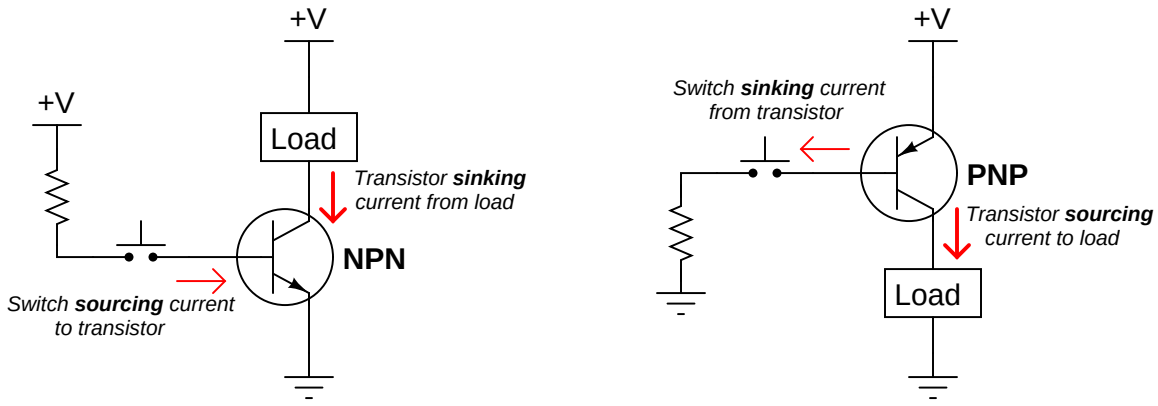
Answer 38



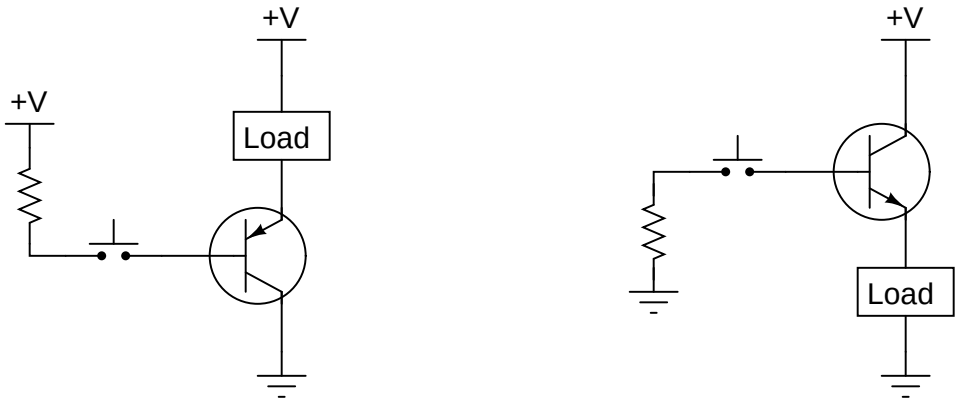
Follow-up question: explain why neither of the following transistor circuits will work. When the pushbutton switch is actuated, the load remains de-energized:



Answer 39



Follow-up question: explain why neither of the following transistor circuits will work. When the pushbutton switch is actuated, the load remains de-energized:



Answer 40

Switch A	Switch B	Light Bulb
Unpressed	Unpressed	Off
Unpressed	Pressed	Off
Pressed	Unpressed	On
Pressed	Pressed	On

Note how Switch B has no effect on the PLC's output status! The reason for this is the placement of the two identically-addresses coils in the PLC program: each rung writes either a 0 or a 1 to the same output bit `Q0.1`, but only the *last* rung's state is in effect when the PLC finishes its scan of the program and updates the output registers to actually turn its output channels on or off.

This is why it is a bad idea to assign the same address to multiple coils in a PLC program, the only exception to this rule being when the coils in question are retentive (i.e. "Set" and "Reset" or "Latch" and "Unlatch" coils) in which case complementary coil pairs bearing the same address is proper. Regular, non-retentive coil instructions, however, will conflict with one another in a PLC program if they bear the same bit address.

Answer 41

Hint: to identify whether an I/O point is sourcing or sinking, sketch arrows showing the direction of electric current (using conventional flow notation) where wires connect to the I/O channel terminals. If the arrow shows current *exiting* the PLC channel and headed toward an external device, then that I/O channel is *sourcing* current to that device. If the arrow shows current *entering* the PLC channel from an external device, then that I/O channel is *sinking* current from that device.

Answer 42

Partial answer:

- Temperature switch = **cooler than 150 deg F**

Answer 43

Answer 44

Each "wasteful" program uses an output bit as the intermediary bit between the `AND` and `NOT` functions when there is no need.

Answer 45

Partial answer:

Each of the S7-200 counter instructions can count as high as +32767 and as low as −32768. This equates to 16 bits, signed integer (2's complement notation, where the MSB has a negative place-weight value of −32768).

Answer 46

Answer 47

There are no answers provided here! For help, consult the "instruction set" reference manual for your PLC, which will describe in detail how each type of instruction is supposed to function in your PLC.

Answer 48

Temperature = below 135 ^{o}F

Level = above 23 inches

Pressure = below 17 PSI

Answer 49

If the lamp is energized, we know that the top two virtual contacts (`X1` and `X2`) are colored, and/or the bottom two virtual contacts (`X3` and `X2`) are colored.

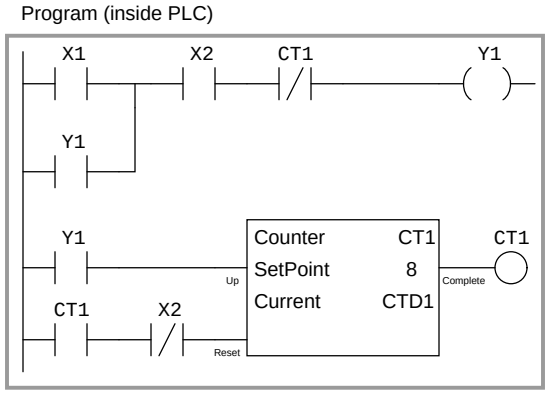
For the top two virtual contacts to be colored, `X1` must be 0 and `X2` must be 1. This equates to a pressure less than 32 PSI and a level less than 10 inches.

For the bottom two virtual contacts to be colored, `X3` must be 1 and `X2` must be 0. This equates to a temperature greater than 99 ^{o}F and a level greater than 10 inches.

Answer 50

This PLC program allows the motor to start up *7* times. If you thought the correct number of start-ups was eight, consider the fact that the counter's output bit (`CT1`) gets set when the counter's current value *equals* the SetPoint value, not when it *exceeds* the SetPoint value.

Here is a solution for an alternative Reset function:



In order to reset the counter, the operator must press the Stop button (after the counter has disabled the system from starting).

Answer 51

Answer 52

Answer 53

Answer 54

Answer 55

Answer 56

Answer 57

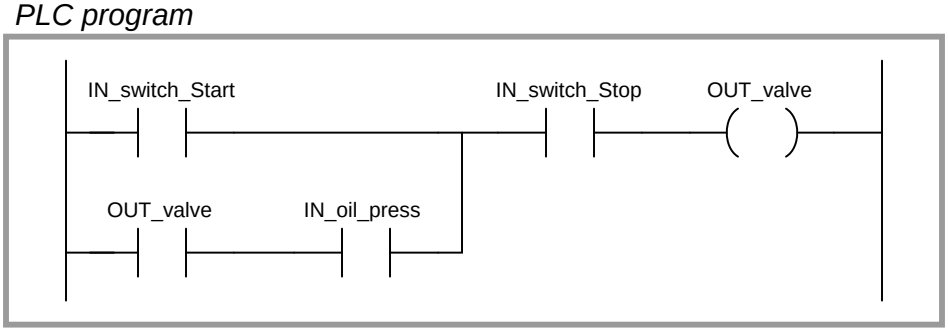
Answer 58

Answer 59

Answer 60

Answer 61

This is one possible fix for the problem:



Answer 62

Answer 63

Hint: the "P" contact instructions are *positive transition* instructions, "activating" whenever their respective bits transition from 0 to 1, but returning to an "inactive" state whenever the bit value holds at either 0 or 1.

Answer 64

Answer 65

Answer 66

Hint: the contact address `C5:0.ACC/13` refers to the 13th bit of the counter's accumulator register, which is a 16-bit binary number. The 15th bit would be the MSB, while the 0th bit is the LSB.

Answer 67

Answer 68

Answer 69

Answer 70

Answer 71

Answer 72

Answer 73

Answer 74

Answer 75

Answer 76

Answer 77

Answer 78

Answer 79

Answer 80

Answer 81

This is a graded question – no answers or hints given!

Answer 82

This is a graded question – no answers or hints given!

Answer 83

This is a graded question – no answers or hints given!

Answer 84

This is a graded question – no answers or hints given!

Answer 85

This is a graded question – no answers or hints given!

Answer 86

This is a graded question – no answers or hints given!

Answer 87

This is a graded question – no answers or hints given!

Answer 88

This is a graded question – no answers or hints given!

Answer 89

This is a graded question – no answers or hints given!

Answer 90

This is a graded question – no answers or hints given!

Answer 91

Answer 92

Your loop diagram will be validated when the instructor inspects the loop with you and the rest of your team.